Universidad Carlos iii de Madrid

Escuela Politécnica Superior

Ingeniería superior en Informática

PROYECTO DE FIN DE CARRERA

Desarrollo y evaluación de un algoritmo de ponderación local de características en aprendizaje relacional basado en el prototipo más cercano

Autor: Ruben Suarez Diez
Tutor: Dr. Fernando Fernandez Rebollo

Septiembre 2010

TÍTULO: *Desarrollo y evaluación de un algoritmo de ponderación local de características en aprendizaje relacional basado en el prototipo más cercano*

AUTOR: *Rubén Suárez Díez*

TUTOR: *Fernando Fernández Rebollo*

La defensa del presente Proyecto Fin de Carrera se realiza el día _____, siendo calificada por el siguiente tribunal:

PRESIDENTE	
SECRETARIO	
VOCAL	

Habiendo obtenido la siguiente calificación:

CALIFICACIÓN	

Presidente **Secretario** **Vocal**

Agradecimientos

Querría agradecer a mi tutor del proyecto, el Dr. Fernando Fernández, su atención y ayuda a lo largo del desarrollo de este trabajo. También me gustaría agradecer a Rocío García Durán el esfuerzo que dedicó a este proyecto en sus inicios.

Me gustaría agradecer a los amigos y compañeros que he conocido estos años en la Universidad los buenos momentos que pasé con ellos. Espero mantener su amistad durante mucho tiempo.

Agradezco a mi familia su cariño y afecto todos estos años, especialmente a mis padres. Quiero agradecer a mi madre, María Teresa, todo su esfuerzo a lo largo de mi vida para que pudiese llegar hasta aquí. Por último me gustaría agradecer a mi padre, Nemesio, la disposición para apoyarme que siempre tuvo, bajo cualquier circunstancia.

Índice general

1. Introducción **1**

1.1. Motivación 3

1.2. Objetivos del proyecto 5

1.3. Organización del documento 6

2. Estado de la cuestión **7**

2.1. Aprendizaje Automático 7

2.1.1. Aprendizaje Supervisado 8

2.1.2. Algoritmos basados en instancias 10

K-NN 10

K-Medias 13

LVQ 14

2.2. ENPC 15

2.3. LFW-NPC 21

2.3.1. Ponderación local de características 21

2.3.2. Cálculo de la distorsión y pesos asociados a cada región 21

2.4. Aprendizaje Relacional 23

2.4.1. Representación relacional de dominios 24

2.4.2. Aproximaciones al aprendizaje automático relacional 26

2.4.3. Aproximaciones relacionales basadas en instancias 27

2.5. RNPC 29

2.6. Weka y Relational Weka 31

2.7. Evaluación de resultados 34

3. LFW-RNPC **37**

3.1. Flujo de ejecución 38

3.2. Cálculo del medoide 40

3.3. Medida de distancia 42

3.3.1. RIBL 42

3.3.2. Distancia LFW-RNPC 44

3.4. Cálculo de distorsión y actualización de pesos 48

3.5. Parámetros 49

4. Experimentación **51**

4.1. Método de evaluación 51

4.2. Configuración de los experimentos 52

4.3. Dominios artificiales 53

4.3.1. Dominio 1 53

4.3.2. Dominio 2 56

4.3.3. Dominio 3 59

4.4. Dominios Experimentales 61

4.4.1. Musk 61

4.4.2. Diterpenes 63

4.4.3. Mutagenesis 65

4.5. Dominios de planificación 67

4.5.1. Blocksworld 67

4.5.2. Zenotravel 69

4.6. Resumen de resultados 71

5. Conclusiones y trabajo futuro **75**

5.1. Conclusiones sobre los objetivos iniciales del proyecto 75

5.2. Conclusiones sobre la ponderación de pesos en dominios relacionales . 77

5.3. Conclusiones sobre los resultados experimentales 78

5.4. Trabajo futuro 79

5.4.1. Distancia y cálculo de la distorsión 79

5.4.2. Representante de la región 79
5.4.3. Optimización del algoritmo 80
5.4.4. Flujo de ejecución 80
5.5. Conclusiones generales 80

A. Diseño y desarrollo **81**
A.1. Requisitos de la implementación 81
A.1.1. Selección de tecnología para el desarrollo 81
A.1.2. Generación y obtención de dominios 82
A.1.3. Gestión de ejemplos 82
A.1.4. RNPC 83
A.1.5. Optimización del tiempo de ejecución 83
A.1.6. Evaluación de resultados 84
A.1.7. Registro de resultados 84
A.1.8. Gestión de parámetros del algoritmo 85
A.2. Diseño 87
A.2.1. Diseño de componentes 87
A.2.2. Diseño de clases 88
Componente EsquemaRelacional 88
Componente LFW-RNPC 89
Componente Principal 90

B. Gestión del proyecto **91**
B.1. Planificación 91
B.2. Recursos Humanos 93
B.3. Recursos materiales 93

Bibliografía **95**

Lista de Figuras

2.1. Clasificación en k-NN 12
2.2. Clasificación en K-NN con $k = 3$ y $k = 5$ 13
2.3. Diagrama de Voronoi de 3 regiones 14
2.4. Ejemplo de población de prototipos en LVQ 15
2.5. Diagrama de flujo del algoritmo ENPC 16
2.6. Mutación del prototipo 17
2.7. Ejemplo de reproducción 18
2.8. Ejemplo de lucha cooperativa 19
2.9. Ejemplo de lucha competitiva 19
2.10. Ejemplo de movimiento 20
2.11. Ejemplo de instancia relacional 25
2.12. Ejemplo de esquema relacional 26
2.13. Diferencia entre medoide y centroide 31
2.14. Dependencia entre $relacion1$ y $relacion2$ 33
2.15. Iteración k de una validación cruzada 35

3.1. Diagrama de flujo de LFW-RNPC 38
3.2. Ejemplo de matriz de distancias de la región 41
3.3. Ejemplo de esquema relacional 42

4.1. Gráficas de ejecución, $z = 0{,}0$, dominio Dominio 1 54
4.2. Gráficas de ejecución, $z = 0{,}5$, dominio Dominio 1 54
4.3. Gráficas de ejecución, $z = 1{,}0$, dominio Dominio 1 54
4.4. Gráficas de ejecución, $z = 0{,}0$, dominio Dominio 2 57

4.5. Gráficas de ejecución, $z = 0{,}5$, dominio Dominio 2 57
4.6. Gráficas de ejecución, $z = 1{,}0$, dominio Dominio 2 57
4.7. Gráficas de ejecución, $z = 0{,}0$, dominio Dominio 3 60
4.8. Gráficas de ejecución, $z = 0{,}5$, dominio Dominio 3 60
4.9. Gráficas de ejecución, $z = 1{,}0$, dominio Dominio 3 60
4.10. Gráficas de ejecución, $z = 0{,}0$, dominio Musk 62
4.11. Gráficas de ejecución, $z = 0{,}5$, dominio Musk 62
4.12. Gráficas de ejecución, $z = 1{,}0$, dominio Musk 62
4.13. Gráficas de ejecución, $z = 0{,}0$, dominio Diterpenes 64
4.14. Gráficas de ejecución, $z = 0{,}5$, dominio Diterpenes 64
4.15. Gráficas de ejecución, $z = 1{,}0$, dominio Diterpenes 64
4.16. Gráficas de ejecución, $z = 0{,}0$, dominio Mutagenesis 66
4.17. Gráficas de ejecución, $z = 0{,}5$, dominio Mutagenesis 66
4.18. Gráficas de ejecución, $z = 1{,}0$, dominio Mutagenesis 66
4.19. Gráficas de ejecución, $z = 0{,}0$, dominio Blocksworld 68
4.20. Gráficas de ejecución, $z = 0{,}5$, dominio Blocksworld 68
4.21. Gráficas de ejecución, $z = 1{,}0$, dominio Blocksworld 68
4.22. Gráficas de ejecución, $z = 0{,}0$, dominio Zenotravel 70
4.23. Gráficas de ejecución, $z = 0{,}5$, dominio Zenotravel 70
4.24. Gráficas de ejecución, $z = 1{,}0$, dominio Zenotravel 70

A.1. Diagrama de componentes 87
A.2. Clases del componente EsquemaRelacional 88
A.3. Clases del componente LFW-RNPC 89
A.4. Clases del componente principal 90

B.1. Diagrama de Gantt 92

Lista de Tablas

4.1. Resultados en Dominio 1 55
4.2. Resultados en Dominio 2 58
4.3. Resultados en Dominio 3 61
4.4. Resultados en Musk 63
4.5. Resultados en Diterpenes 65
4.6. Resultados en Mutagenesis 65
4.7. Resultados en Blocksworld 69
4.8. Resultados en Zenotravel 71
4.9. Resumen de resultados 73

B.1. Tareas del proyecto 91
B.2. Recursos Humanos 93
B.3. Recursos materiales 93

Capítulo 1

Introducción

Este proyecto de fin de carrera se encuadra en el área de la Inteligencia Artificial. Este término cobró importancia y fue definido en las conferencias de Dartmouth en 1956, donde se establecieron los objetos de estudio de la disciplina a partir de una premisa básica, que se recoge en la propuesta de la conferencia:

> *Proponemos la realización de un estudio de 10 hombres durante dos meses del verano de 1956, en la universidad de Dartmouth en Hanover, New Hampshire. El estudio partirá de la conjetura de que el aprendizaje, o cualquier otro aspecto de la inteligencia, puede en principio ser descrito de una manera suficientemente precisa como para ser simulado en una máquina. Se investigará la posibilidad de hacer que las máquinas usen lenguaje, formen abstracciones y conceptos, resuelvan problemas hasta ahora reservados a los humanos, y puedan mejorarse a sí mismas. Pensamos que un avance significativo puede hacerse en uno o varios de estos problemas si un grupo selecto de científicos trabaja juntos en ello durante un verano.*

Esta propuesta fue llevada a cabo por cuatro ingenieros y científicos muy importantes para el desarrollo de la informática: Claude E. Shannon (fundador de la teoría de la información y el diseño de circuitos digitales), Marvin Minsky (matemático e inventor, premio Turing en el año 1969), N. Rochester (creador del primer ensamblador) y J. McCarthy (inventor del lenguaje LISP y ganador del premio Turing en 1971 por sus contribuciones a la Inteligencia Artificial).

En la propuesta anterior puede observarse como a partir de ese momento ya se realiza un división de las distintas ramas o problemas que comprende el campo, cuyo número se ha ampliado a lo largo de los años por la división de algún área en otras más especializadas y por la adición de otras nuevas que lidian con problemas que no se contemplaron con la suficiente importancia durante los primeros años. Así, entre las áreas más relevantes se pueden nombrar la representación del conocimiento, que tiene como objetivo el desarrollo de formalismos para representar simbólicamente un dominio y facilitar la inferencia; la planificación automática, que investiga los mecanismos para la creación de planes que consigan objetivos determinados; la percepción automática, que desarrolla algoritmos para el análisis de información proveniente de distintos tipos de sensores; y otros.

Este proyecto entra en el área del aprendizaje automático, que definiremos más explícitamente de la siguiente manera [9]: *El campo del aprendizaje automático se preocupa del problema de la construcción de programas que mejoren automáticamente mediante la experiencia.* Aunque durante los primeros años de la Inteligencia Artificial se realizaron diversos esfuerzos en esta área, ésta quedo relegada por otras como la resolución de problemas y la representación del conocimiento. La principal etapa de desarrollo de esta disciplina comenzó en los años 80, y actualmente tiene una gran relevancia a nivel académico e industrial.

Dentro del aprendizaje automático también existen áreas diferentes, cada una con un propósito distinto. Una de las posibles clasificaciones divide el campo en tres áreas distintas, según el problema que se intenta resolver: aprendizaje por refuerzo, donde se pretende aprender una estrategia o política para la consecución de un fin a partir de recompensas asociadas a las posibles acciones; aprendizaje supervisado, donde el objetivo es aprender una relación entre entrada y salida, a partir de ejemplos para los que se conocen estos valores; y por último, aprendizaje no supervisado, donde el objetivo es estructurar u organizar un conjunto de datos, y descubrir información no explícitamente dada en éstos.

Este proyecto se encuadra dentro del aprendizaje supervisado, específicamente en el problema de la clasificación. En este caso la relación que desea aprenderse consiste en la asignación de una etiqueta (o clase) a nuevas observaciones en función de observaciones anteriores para las cuales se conocen las etiquetas asociadas.

Existen varios tipos de algoritmos y modelos de clasificación, con características muy diferentes. Los más relevantes incluyen: los árboles de decisión, estructuras de datos inducidas a partir de observaciones anteriores que infieren la clase de un ejemplo

a partir de una serie de preguntas sobre la observación a clasificar; las redes de neuronas artificiales, modelos numéricos que intentan imitar el comportamiento de las redes de neuronas biológicas; las máquinas de vector soporte, que realizan la clasificación de observaciones a partir del cálculo de hiperplanos discriminantes que dividen el conjunto de datos entre las distintas clases; los basados en el teorema de Bayes, como Naive Bayes o las redes bayesianas, que estiman la probabilidad de la pertenencia de una observación a una clase a partir de estimaciones calculadas sobre los datos de entrada; y por último los algoritmos basados en distancias, que utilizan una medida de distancia para determinar la diferencia entre dos observaciones, y clasificar un ejemplo en función de su similitud con otros ejemplos conocidos.

Al igual que en otros campos de la Inteligencia Artificial, la representación del dominio del problema que desea resolverse es un aspecto muy importante. La representación más estudiada en clasificación es la basada en pares atributo-valor, o proposicional, donde las observaciones se representan como tuplas en una tabla. El inconveniente de esta representación es que existen dominios cuya representación proposicional es muy complicada, ineficiente, o directamente imposible. Ejemplos de dominios donde se da esta dificultad son aquellos que describen compuestos químicos, estructuras gramaticales, estados de un dominio de planificación, y en general cualquier dominio donde la estructura de cada observación sea variable.

Por lo tanto, en estos dominios se requiere de un lenguaje de representación más expresivo, que puede conseguirse mediante un tipo de representación basada en lógica de primer orden, o relacional. El aprendizaje relacional es el área del aprendizaje automático que estudia los problemas que requieren de este tipo de representación, siendo el algoritmo descrito en este proyecto uno de ellos.

En las siguientes secciones de este capítulo se especifican de manera detallada la motivación y los objetivos de este proyecto.

1.1. Motivación

En las aproximaciones de aprendizaje supervisado basado en instancias la asignación de pesos a las características que describen el dominio, con el propósito de mejorar alguna medida de calidad de los resultados, es un método bien conocido y aplicado en diversos algoritmos.

La asignación de estos pesos puede hacerse manualmente o a través de alguna

técnica que establezca el valor de cada peso en función de los datos de entrada del algoritmo. Esta asignación de pesos puede ser global, donde cada característica tiene un peso asociado independientemente de las instancias particulares que se estén comparando, o local, donde el valor del peso asociado a cada característica depende de las instancias concretas que se comparen.

En aprendizaje relacional existen varias medidas de distancias que utilizan pesos asociados a las características del dominio. Los principales inconvenientes en estos casos es que normalmente los pesos se establecen a mano (dependiendo del conocimiento del dominio del usuario), y que esta ponderación es global.

Existen algoritmos en aprendizaje proposicional que realizan automáticamente esta ponderación. En este proyecto los algoritmos de referencia son LFW-NPC (*Local Feature Weighting Relational Nearest Prototype Classifier*), un modelo proposicional de clasificación supervisada que actualiza localmente los pesos asociados a los atributos, y que es una ampliación de ENPC (*Evolutionnary Nearest Prototype Classifier*), un algoritmo de clasificación evolutivo. También consideraremos el algoritmo RNPC (*Relational Nearest Prototype Classifier*) un algoritmo de clasificación supervisada para dominios relacionales.

Por lo tanto parece razonable establecer la hipótesis de que la consecución de una ponderación automática de características (dependiente de los datos de entrada) local (adaptada a cada región del espacio de instancias) pueda mejorar de alguna manera las aproximaciones anteriores de ponderación de atributos en algoritmos de aprendizaje relacional basados en distancias.

Para probar la hipótesis anterior presentamos el algoritmo LFW-RNPC (Local Feature Weighting Relational Nearest Prototype Classifier), que actualiza localmente los pesos asociados a los atributos de un dominio relacional.

Para ello utilizaremos la plataforma Weka, que implementa varios algoritmos de aprendizaje automático, y Relational Weka, una extensión de la plataforma que define formatos de representación relacional y varias medidas de distancia relacionales.

1.2. Objetivos del proyecto

En este apartado se describen y enumeran los principales objetivos y subobjetivos que definen el alcance del proyecto.

1. Diseño e implementación del algoritmo LFW-RNPC

   a) *Estado de la cuestión*
   En esta fase deben revisarse los algoritmos más relevantes para este proyecto, lo que comprende la lectura y comprensión de la literatura (artículos y libros) que cubra el aprendizaje basado en instancias, aprendizaje relacional y los algoritmos ENPC, RNPC y LFW-NPC.

   b) *Estudio de las APIs de Weka y Relational Weka, y de la implementación del algoritmo RNPC*
   Ya que el desarrollo del proyecto se basará en los servicios proporcionados por la herramienta Relational Weka para la representación de dominios relacionales de aprendizaje automático, se necesita estudiar las APIs ofertadas en este paquete de software y establecer que componentes se necesita implementar y cuáles pueden reutilizarse. También se estudia la implementación existente del algoritmo RNPC.

   c) *Diseño de la medida de distancia relacional*
   Estudio de las diversas medidas de distancias relacionales en la literatura, y diseño de la distancia o distancias que utilizaremos en la versión final del algoritmo.

   d) *Diseño y elección del método de actualización de pesos*
   Estudio de la técnica de actualización de pesos y adaptación a las medida de distancia utilizada.

   e) *Diseño de componentes*
   Estructuración y división de las clases que utiliza el algoritmo para gestión de los datos de entrada y de las clases propias de la ejecución del algoritmo, así como la determinación de los métodos algorítmicos utilizados para resolver las tareas particulares requeridas en el flujo de ejecución del algoritmo.

   f) *Implementación del algoritmo*
   Implementación del sistema software de acuerdo la estructura definida en el diseño de los componentes del algoritmo.

   g) *Validación y verificación del algoritmo*
   Comprobación de la corrección del código implementado.

2. Experimentación y evaluación de LFW-RNPC

   a) *Definición y selección de dominios de experimentación*
   Creación de dominios de prueba propios para observar la diferencia de resultados respecto a la versión con pesos y la versión sin pesos. Elección de dominios existentes para la comparativa de resultados.

   b) *Experimentación*
   Ejecución del algoritmo con distintos grados de actualización de pesos en los dominios seleccionados.

   c) *Análisis de los resultados*
   Conclusiones sobre los resultados obtenidos en la experimentación, y reflexiones sobre el trabajo futuro.

3. Redacción de la memoria del proyecto
   Plasmar en un documento el trabajo realizado.

1.3. Organización del documento

Esta memoria se organiza de la siguiente manera. En el segundo capítulo se presenta el estado de la cuestión, donde se definen los conceptos fundamentales para este proyecto y se describen los algoritmos fundamentales basados en distancias aplicados a dominios relacionales. En el tercer capítulo se presenta el algoritmo LFW-RNPC, explicando las decisiones de diseño y las diferencias con otros algoritmos. En el cuarto capítulo se describe el método de experimentación, se muestran los resultados obtenidos y se resumen las conclusiones de éstos. En el quinto capítulo se dan las conclusiones finales sobre el trabajo realizado en el proyecto, y se comentan las posibles líneas de trabajo futuro. Por último en los anexos se adjuntan detalles del proyecto no contemplados en otros capítulos de la memoria.

Capítulo 2

Estado de la cuestión

En este capítulo se resumen conceptos fundamentales del aprendizaje automático, y se describen los principales algoritmos relacionados con este proyecto.

En la primera sección se amplian los fundamentos de aprendizaje automático que se dieron en el primer capítulo, clasificando las distintas aproximaciones existentes y describiendo las más relevantes para este proyecto. En la segunda sección se explica el algoritmo ENPC, que es la base para del algoritmo LFW-NPC, que incorpora la ponderación de caraterísticas que este proyecto desarrolla para el caso relacional, y que se explica en la tercera sección. En la cuarta sección se comentan los aspectos más relevantes del aprendizaje relacional. En la sección quinta se explica la adaptación de ENPC al caso relacional, RNPC. En la sección 6 se describen la platafora WEKA y su extensión Relational WEKA. Por último, en la sección 7, se explican los métodos habituales de evaluación de resultados en problemas de clasificación.

2.1. Aprendizaje Automático

Como se comentó en el primer capítulo el objetivo del aprendizaje automático es la creación de programas que sean capaces de mejorar con la experiencia. De acuerdo a cómo sea la experiencia de la que se aprende, el objetivo y la representación utilizada pueden clasificarse de distintas maneras los algoritmos existentes.

La experiencia de la que se va a aprender puede consistir en una serie de observaciones previas o en la observación directa del entorno. Si se aprende de un conjunto de observaciones previas, entonces el aprendizaje es *offline*. Si el aprendizaje se realiza

mediante el procesado continuo de observaciones individuales entonces es *online*.

Respecto al objetivo del aprendizaje, las categorías más relevantes incluyen el aprendizaje supervisado (donde se dispone de alguna información previa para comprobar el éxito del algoritmo y se pretende averiguar una relación entre las instancias de aprendizaje y un valor discreto o continuo concreto), aprendizaje no supervisado (dónde no se dispone información suficiente para contrastar los resultados del algoritmo y el objetivo es descubrir información no explícita en los datos de entrada) y aprendizaje por refuerzo (donde el aprendizaje se basa en la observación del entorno, y en la retroalimentación obtenida en respuesta a una acción).

Otro criterio importante para clasificar un algoritmo de aprendizaje automático es el modo de representación que utiliza para describir el dominio del problema. Cuando el dominio se compone de una única relación, donde cada instancia se corresponde con un vector de valores, decimos que es un dominio proposicional. En cambio si para la representación de cada instancias es necesario utilizar varios predicados, los cuales pueden hacer referencia a otros predicados, decimos que es un dominio relacional. Esta diferencia de representación es muy importante, ya que existen varios dominios en los cuales es necesario utilizar una representación relacional. El paso de un modo de representación a otro no es trivial, y actualmente existen varias líneas de investigación que estudian exclusivamente el aprendizaje relacional y los problemas que plantea respecto al aprendizaje proposicional.

Respecto a los criterios anteriores el algoritmo presentado en este proyecto (LFW-RNPC) es un algoritmo offline, supervisado y relacional.

2.1.1. Aprendizaje Supervisado

En el aprendizaje supervisado se posee información para contrastar el éxito del algoritmo en la asignación de un valor determinado a cada instancia del dominio. Los problemas asociados al aprendizaje supervisado son dos, principalmente: clasificación, donde se debe asignar una etiqueta de un conjunto discreto y finito a cada instancia, y regresión donde se asigna un valor numérico (generalmente un número real). A continuación se enumeran las principales aproximaciones al aprendizaje supervisado:

- Basados en árboles de decisión: estos algoritmos analizan un conjunto de datos para construir un árbol de clasificación o regresión. Una vez construido el árbol, cuando se evalúa un ejemplo, se recorren las ramas del árbol, donde cada rama representa una condición sobre alguna de las características que describen cada instancia. En la hoja del árbol se encuentra la clase asignada a la instancia en el caso de la clasificación, o alguna fórmula que determine el valor numérico asociado en el caso de la regresión. Los algoritmos actuales más relevantes para el aprendizaje de árboles de decisión son J48 y C4.5.

- Basados en redes de neuronas: en estos algoritmos se aprende un conjunto de pesos que describe la arquitectura de una red de neuronas. Cada neurona individual tiene asociada una función de activación, que determina el valor de su salida hacia otras neuronas en función de su entrada. Los modelos más relevantes en aprendizaje supervisado son los perceptrones multicapa y las redes de neuronas de base radial.

- Basados en máquinas de vectores de soporte: en estos algoritmos los datos de entrada son vistos como vectores de números de tantas dimensiones como características existan en el dominio. La separación en categorías se realiza mediante la construcción de un hiperplano que separa de forma óptima los datos según algún criterio de calidad.

- Basados en el teorema de Bayes: esta aproximación se basa en el cálculo de la probabilidad de la pertenencia de un ejemplo a una clase, mediante inferencia bayesiana. Existen dos aproximaciones principales: *Naive Bayes*, que asume la independencia mutua de todas las características que describen el dominio, y calcula las posibilidades a prioiri necesarias para hacer la inferencia a partir de los datos de entrada; y las redes bayesianas, grafos que expresan distribuciones conjuntas de probabilidad de una manera compacta.

- Basados en instancias: estas aproximaciones se basan en una comparación con instancias conocidas anteriormente para la clasificación.

  El algoritmo presentado en este proyecto se encuentra dentro de la última categoría, por lo que esta aproximación se explica en más detalle en el siguiente apartado.

2.1.2. Algoritmos basados en instancias

Los algoritmos basados en instancias se basan en la comparación entre instancias conocidas y las nuevas instancias a clasificar, a diferencia de otros algoritmos que contruyen un modelo a partir de los ejemplos de entrenamiento, y consultan el modelo construido con las características de la nueva instancia para clasificarla.

Para hacer la comparación entre instancias conocidas y desconocidas debe establecerse alguna medida de similitud, que suele basarse en una relación de distancia, definida en función de las características del dominio. Los algoritmos basados en instancias más conocidos son K-NN (*K Nearest Neighbours*, o K vecinos más cercanos), K-medias y LVQ.

K vecinos más cercanos

Este es el algoritmo más sencillo de los basados en instancias. En la fase de entrenamiento almacena todos los ejemplos de entrada, y clasifica nuevas instancias encontrando los k vecinos más cercanos (los k ejemplos conocidos que minimizan la distancia al nuevo ejemplo). Asigna la clase de salida como una combinación de las clases de estos vecinos (asignando la clase mayoritaria, o ponderando de alguna manera la importancia de cada vecino).

La elección de la medida de distancia es un aspecto muy importante. Esta elección suele hacerse en función del conjunto de datos que vaya a analizarse. Entre las más utilizadas se encuentran:

- Distancia de bloque: esta distancia considera la diferencia absoluta entre valores.

$$d(\vec{x}, \vec{y}) = \sum |x[i] - y[i]| \tag{2.1}$$

- Distancia euclídea: esta es la distancia más conocida. Mide la distancia entre 2 puntos en un espacio euclídeo.

$$d(\vec{x}, \vec{y}) = \sqrt{\sum_i |x[i] - y[i]|^2} \tag{2.2}$$

- Distancia euclídea ponderada: cálcula la distancia euclídea asignando a cada

dimensión una importancia relativa a un peso $w[i]$.

$$d(\vec{x},\vec{y}) = \sqrt{\sum_i w[i]|x[i]-y[i]|^2} \tag{2.3}$$

- Distancia de Chebysev: mide la máxima diferencia en alguna de las dimensiones.

$$d(\vec{x},\vec{y}) = \max_i(|\vec{x}[i]-\vec{y}[i]|) \tag{2.4}$$

- Distancia de Minkowski: las distacias de bloque ($p=1$), euclídea ($p=2$) y de Chebysev($p=\infty$) son casos particulares de esta distancia:

$$d(\vec{x},\vec{y}) = \sqrt[p]{\sum |x[i]-y[i]|^p} \tag{2.5}$$

- Distancia de Mahalanobis: esta métrica es muy utilizada en estadística. S es una matriz de covarianzas, que permite modificar la importancia relativa de cada dimensión en el resultado.

$$d(\vec{x},\vec{y}) = \sqrt{(\vec{x}-\vec{y})^T S(\vec{x}-\vec{y})} \tag{2.6}$$

Como puede observarse en los ejemplos anteriores, cada distancia se define respecto a dos vectores de números reales, pero en la aplicación de métricas de distancia a problemas de aprendizaje automático deben definirse además medidas de diferencia entre valores no numéricos, ya que es posible que la descripción del dominio contenga características que no puedan describirse numéricamente. Por ejemplo, es frecuente que en la descripción de dominios existan atributos que sólo admiten valores de un conjunto discreto finito de posibilidades (nominales). A menos de que se disponga de conocimiento del dominio que permita establecer alguna similitud entre los valores normalmente sólo se tiene en cuenta si estos son distintos o no:

$$a,b \in V, |V| \in \mathbb{N}, d(a,b) = \begin{cases} 1 & \text{si} \quad a \neq b \\ 0 & \text{si} \quad a = b \end{cases}$$

También existen varios criterios de diferencia para cadenas de caracteres y listas, desde el número de caracteres distintos, comparación entre el valor o carácter en la misma posición, o el número de operaciones para convertir una cadena o lista en otra [13].

Se presenta una ejemplo gráfico en la figura 2.1, donde se desea clasificar instancia en un dominio con 2 clases (A y B) con $k = 5$. La cruz central representa la instancia que se va a clasificar, mientras los círculos representan instancias conocidas de la clase B y los cuadrados instancias conocidas de la clase A. Si asumimos que el modo de asignar la clase se basa en el voto mayoritario de los vecinos, se asignaría la clase A, ya que existen más ejemplos de la clase A que de la clase B.

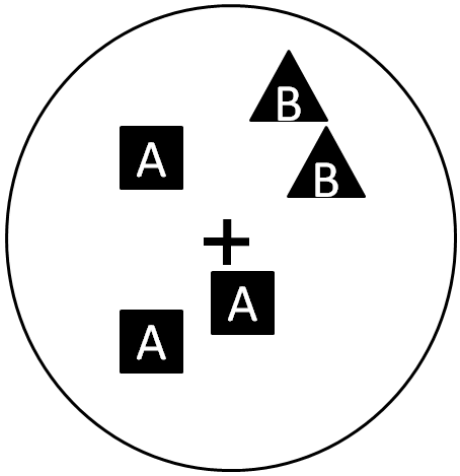


Figura 2.1: Clasificación en k-NN

Un inconveniente importante del algoritmo es que los resultados obtenidos son muy sensibles a la elección del parámetro k y de la medida de distancia utilizada. La diferencia en la clasificación se muestra gráficamente en la figura 2.2. En este ejemplo para un $k = 3$ la clase asignada es A, mientras que para $k = 5$ la clase asignada es B.

Generalmente el valor de k se elige mediante mediante la comparación de los resultados obtenidos con distintos valores, ya que el mejor valor posible depende del conjunto de datos que se esté utilizando.

Otro inconveniente de este algoritmo es que tiene un coste muy alto en memoria y en tiempo de clasificación para conjuntos de datos grandes, ya que es necesario almacenar todas las instancias de entrenamiento, y cuando se va a clasificar una nueva instancia deben compararse todas las instancias con la nueva, para encontrar los vecinos más cercanos.

Los algoritmos basados en prototipos resuelven algunos de los problemas planteados en en k-NN. Este tipo de algoritmos genera un conjunto reducido de instancias

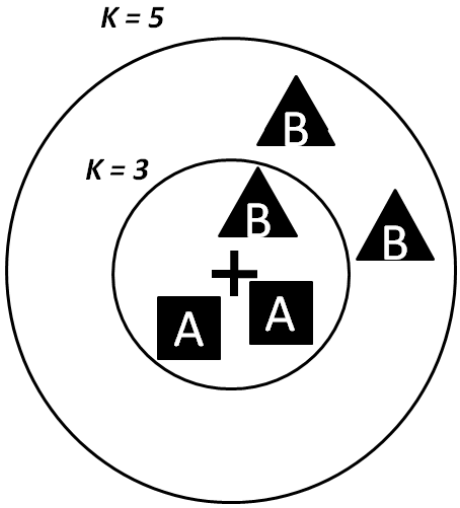


Figura 2.2: Clasificación en K-NN con $k = 3$ y $k = 5$

(los prototipos) que se utilizan para comparar las nuevas instancias, asignando a cada nueva instancia la clase del prototipo más cercano.

K-Medias

K-medias no es un algoritmo de clasificación, si no un algoritmo de agrupamiento o *clustering*, pero es importante para explicar los algoritmos de clasificación basados en prototipos.

A partir de un parámetro k, el algoritmo agrupará los datos de entrada en k grupos o *clusters*, alrededor de k medias, definiendo una región de Voronoi propia cada una de ellas.

Un diagrama de Voronoi es una descomposición de un espacio respecto a un conjunto discreto de puntos. Cada región de Voronoi es el subconjunto de puntos más cercanos a uno de los puntos de entrada, o centroides. En la figura 2.3 se muestra un diagrama de Voronoi de 3 regiones.

El algoritmo sigue los siguientes pasos:

1. Las medias se posicionan en la misma posición de k ejemplos conocidos, aleatoriamente seleccionados.

2. Se calculan las regiones de Voronoi asociadas a cada prototipo.

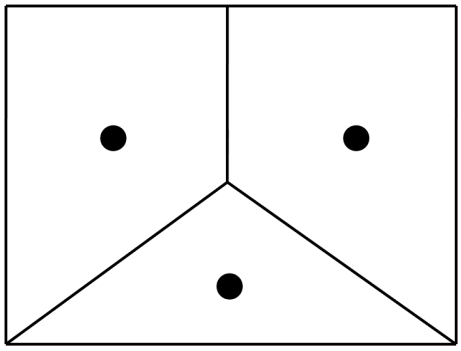


Figura 2.3: Diagrama de Voronoi de 3 regiones

3. Cada media se sitúa en el centroide de su región (el punto equidistante a todos los puntos contenidos en la región).

4. Se comprueba el criterio de finalización, y en caso de que no se cumpla se vuelve al paso 2.

Como criterio de finalización es frecuente utilizar alguna medida de estabilidad de la posición de los prototipos (por ejemplo la diferencia de posición entre iteraciones).

En este algoritmo normalmente el establecimiento del parámetro k se realiza mediante prueba y error, probando distintas configuraciones con un mismo conjunto de datos y comparando los resultados, ya que no se puede saber a priori el número de *clusters*, o grupos, más adecuado para el problema.

Learning Vector Quantization

Learning Vector Quantization [15] (LVQ) es un algoritmo de clasificación supervisada basado en mapas autoorganizativos. El algoritmo determina un conjunto de prototipos para clasificar un conjunto de datos de entrada etiquetados en las distintas clases.

El algoritmo parte de una población inicial de prototipos (cuyo número se fija como parámetro). Cada prototipo define una región de Voronoi propia, y su clase es la mayoritaria entre las instancias comprendidas en ésta. A lo largo de las iteraciones del algoritmo estos prototipos se resitúan en el espacio de los datos de entrada (ajustando un vector de pesos asociado a cada prototipo), hasta que se llega a una determinada estabilidad de la población de prototipos.

En la figura 2.4 se muestra un ejemplo de la configuración de clasificadores producida por LVQ. En la figura se pueden observar los vecinos de cada prototipo (unidos por una línea discontinua), las fronteras de las regiones de Voronoi (las líneas continuas delgadas) y la frontera de decisión entre clases (las líneas continuas oscuras).

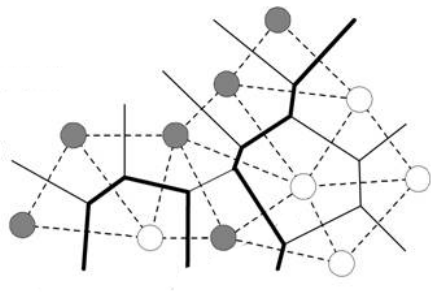


Figura 2.4: Ejemplo de población de prototipos en LVQ

2.2. Evolutionnary Nearest Prototype Classifier

ENPC [2] es un algoritmo proposicional de clasificación supervisada, basado en prototipos. Se inspira en un ecosistema donde varios seres vivos cohabitan, se reproducen, se alimentan, luchan entre sí y mueren.

En este algoritmo evolutivo, los individuos de la problación son los prototipos que componen el clasificador. Cada prototipo define una región (un conjunto de instancias asociadas) y es caracterizado por un conjunto de atributos que definen su calidad:

1. *Expectación*: esta característica mide la proporción de instancias de la misma clase del prototipo que se esperan que clasifique correctamente. En la siguiente ecuación $||S_j||$ representa el total de instancias de la clase del prototipo en los datos de entrada y $regiones(s_j)$ representa el número de regiones o prototipos existentes de su misma clase en la población actual.

$$expectacion(p_i) = \frac{||S_j||}{regiones(s_j)} \tag{2.7}$$

2. *Precisión*: es la relación entre el número de instancias de la misma clase del prototipo(V_{ij}), respecto al total de instancias de su región(T_i).

$$precision(p_i) = \frac{V_{ij}}{T_i} \tag{2.8}$$

3. *Aportación*: esta característica representa la proporción de instancias de la clase cubiertas por el prototipo (V_ij).

$$aportacion(p_i) = \frac{V_{ij}}{\frac{expectacion(p_i)}{2}} \tag{2.9}$$

4. *Calidad*: la calidad de la clase se calcula teniendo en cuenta la precisión y la aportación del prototipo.

$$calidad(p_i) = min(1, precision(p_i) * aportacion(p_i)) \tag{2.10}$$

La entrada del algoritmo consiste en los ejemplos de entrenamiento clasificados, y la salida consistirá en un conjunto de prototipos, a partir de los cuáles se podrán clasificar nuevas instancias mediante la asignación de la clase del prototipo más cercano.

A continuación se describen los pasos que sigue el algoritmo:

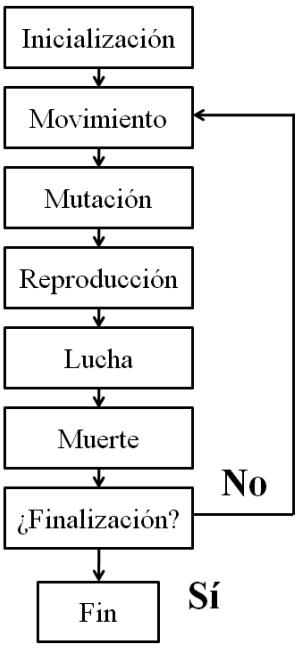


Figura 2.5: Diagrama de flujo del algoritmo ENPC

1. *Inicialización*:
   A diferencia de otros algoritmos, ENPC no necesita que se especifiquen parámetros. El prototipo inicial se elige aleatoriamente entre todas las instancias de

entrenamiento. La región de este primer prototipo contiene a todas las instancias de entrenamiento. A partir de los operadores que se describen a continuación se determinará la creación y eliminación de nuevos prototipos, así como el traspaso de instancias entre diferentes regiones. El número de prototipos se regula automáticamente en función de las características del clasificador.

2. *Actualización de información*:
   En este paso se calculan las características de todos los prototipos presentadas anteriormente. Además se reasignan las instancias de cada región, situando cada una en la región del prototipo más cercano.

3. *Mutación*:
   El objetivo de este operador es etiquetar a cada clasificador con la clase mayoritaria de su región, como se puede observar en la figura 2.6. El prototipo de la clase 2 se convierte a la clase 1, ya que existen más instancias de la clase 1 que de la clase 2 en su región.

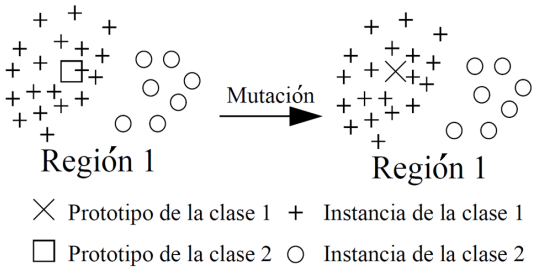


Figura 2.6: Mutación del prototipo

4. *Reproducción*
   El objetivo de este operador la creación de nuevos prototipos, de tal manera que el prototipo que se reproduce pueda aumentar su caldidad. Así, la probabilidad de que cada prototipo se reproduzca es dependiente de su calidad:

$$P_{reproduccion}(p_i) = \frac{1 - precision(p_i)}{2} \tag{2.11}$$

   En la figura 2.7 se muestra un ejemplo de reproducción. Un prototipo de la clase 1 engendra un prototipo de la clase 2, ya que el número de instancias de esta clase es muy alta, y esto afecta a su precisión.

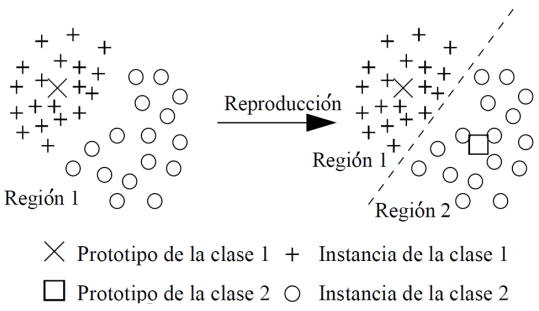


Figura 2.7: Ejemplo de reproducción

5. *Lucha*

   En esta fase los prototipos luchan entre ellos para traspasar instancias de una región a otra. Cada prototipo decide si lucha o no con alguno de sus vecinos. Los vecinos de los prototipos se asignan durante la asignación de instancias y son aquellos prototipos con la segunda menor distancia a cualquier instancia de la región. Cada prototipo decide si lucha o no con algún vecino en función de la diferencia de calidad que haya entre los dos:

$$P_{lucha}(p_i, p_j) = |calidad(p_i) - calidad(p_j)| \tag{2.12}$$

   Si hay lucha pueden darse dos casos distintos, dependiendo de si los prototipos son de la misma clase o de clase distinta:

   a) *Cooperación*:

   En este caso simplemente el prototipo que inició la lucha recibe las instancias de la región del oponente que sean de su misma clase. En la figura 2.8 se muestra un ejemplo gráfico de este operador. Un prototipo de la clase 1 lucha con otro de la clase 2. Como son de clase distinta, el primer prototipo cede instancias de la clase 2 al otro prototipo.

   b) *Competición*:

   Cuando los prototipos son de la misma clase. En este caso la región a la que se transfieren unos prototipos depende de quién sea el ganador de la lucha. El ganador se decide utilizando una ruleta con dos porciones, cada una proporcional a la calidad de cada prototipo. Además el número de instancias que se transpasan depende de una probabilidad proporcional a la calidad de ambos prototipos. En la figura 2.9 se muestra un ejemplo de

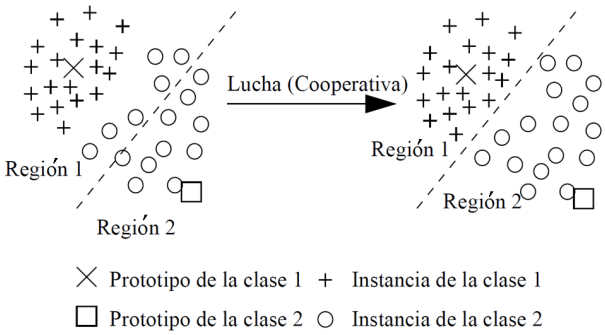


Figura 2.8: Ejemplo de lucha cooperativa

competición. El prototipo de la región 1 gana al prototipo de la región 2, y por eso instancias de este último se traspasan al primero.

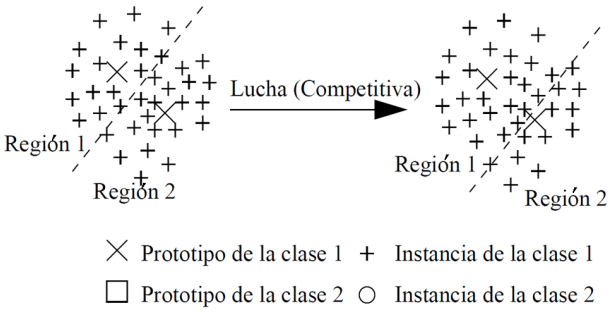


Figura 2.9: Ejemplo de lucha competitiva

6. *Movimiento*
   En esta fase cada prototipo se recoloca como centroide de su región. En la figura 2.10 se muestra como cada prototipo se resitúa en el centro de su región.

7. *Muerte*
   La probabilidad de morir de cada prototipo depende de su calidad, y se rige según la siguiente fórmula:

$$P_{morir}(p_i) = \begin{cases} 0, & \text{si} \quad calidad(p_i) > 0{,}5 \\ 1 - 2 * calidad(p_i) & \text{si} \quad calidad(p_i) < 0{,}5 \end{cases} \tag{2.13}$$

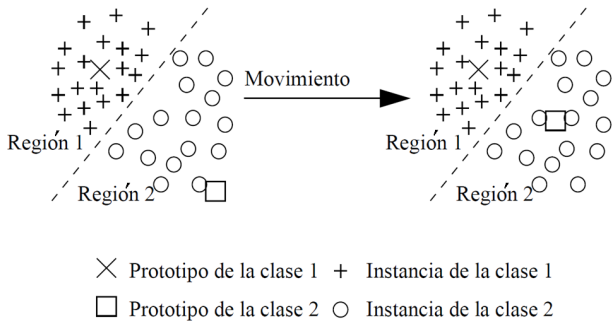


Figura 2.10: Ejemplo de movimiento

El propósito de este operador es eliminar prototipos inútiles de la población (aquellos con poco éxito de clasificación).

8. *Condición de finalización*

El algoritmo acepta varios criterios de finalización, en función del criterio de optimalidad que se utilice. El criterio más usual para un clasificador basado en prototipos se basa en la obtención de un éxito de clasificación alto con el menor número posible de prototipos. Normalmente diferentes ejecuciones del algoritmo evolucionarán hasta converger a un número similar de prototipos y un éxito de clasificación similar, así que el critero de parada puede ser variable. A continuación se enumeran algunos de estos criterios:

a) Número de iteraciones.

b) Éxito de clasificación deseado.

c) Éxito de clasificación deseado y número de iteraciones.

d) Convergencia a un número de prototipos.

e) Convergencia a un éxito de clasificación determinado.

f) Cualquier combinación de los anteriores.

2.3. Local feature weighting in Nearest Prototype Classification

LFW-NPC [3] es una extensión del algoritmo ENPC, que incorpora la ponderación de pesos locales a cada características de la relación, dependientes del prototipo.

2.3.1. Ponderación local de características

En las aproximaciones basadas en la ponderación global de características se aplica normalmente una distancia euclídea ponderada:

$$d_{\vec{w}}(\vec{x},\vec{y}) = \sqrt{\sum_{i=0}^{i<K}(\vec{x}[i]-\vec{y}[i])^2 * \vec{w}[i]} \tag{2.14}$$

donde K es la dimensión del conjunto de datos, x e y son los ejemplos entre los que se calcula la distancia. En este caso el vector w es igual para todo el espacio de instancias.

En LFW-NPC se utiliza un vector de pesos $\vec{w}$ distinto para cada prototipo. La nueva función de distancia se define en la siguiente ecuación:

$$d_{w(\vec{p})}(\vec{x},\vec{p}) = \sqrt{\sum_{i=0}^{i<K}(\vec{x}[i]-\vec{p}[i])^2 * \vec{w_p}[i]} \tag{2.15}$$

Esta nueva medida de distancia tiene la ventaja de que las fronteras entre las regiones de Voronoi definas por las regiones de los prototipos son no lineales.

2.3.2. Cálculo de la distorsión y pesos asociados a cada región

El cálculo de los pesos de cada región consiste en una normalización local de los datos. Esta normalización se basa en la varianza de la localización de las instancias de cada región respecto al centroide de la región. El propósito de esta normalización es que todas las características tengan la misma contribución a la distorsión total.

La distorsión media de una región se define de la siguiente manera:

$$D_{r_i} = \frac{1}{||R_i||} \sum_{\forall \vec{x_i} \in R_i} d^2_{\vec{w_i}}(\vec{x}, \vec{p_i}) \tag{2.16}$$

Donde $||R_i||$ es el número de instancias de la región i y p_i es el prototipo de la región. La distorsión de la región calcula la distancia media de las instancias de la región al representante. Cada atributo contribuirá una determinada cantidad según la varianza de los valores de las instancias de la región respecto a los valores asociados al representante.

El objetivo de la normalización local es que todos los atributos tengan el mismo valor de distorsión:

$$\vec{D_{r_i}}[1] = \vec{D_{r_i}}[2] = ... = \vec{D_{r_i}}[K] \tag{2.17}$$

Para conseguir que esto se impone un peso asociado localmente a cada atributo que aparece en la ecuación (2.14).

A partir de esta igualdad podemos calcular la actualización de los pesos que se realiza en cada iteración.

$$\sum_{\vec{x} \in R_i} (\vec{x}[0] - \vec{p}[0])^2 * \vec{w}[0] = \sum_{\vec{x} \in R_i} (\vec{x}[1] - \vec{p}[1])^2 * \vec{w}[1] \tag{2.18}$$

Para asegurar que se cumpla la igualdad podemos añadir un factor de modificación, como se define en la ecuación 2.19.

$$\sum_{\vec{x} \in R_i} (\vec{x}[0] - \vec{p}[0])^2 * \vec{w}[0] = \sum_{\vec{x} \in R_i} (\vec{x}[1] - \vec{p}[1])^2 * (\vec{w}[1] + \Delta\vec{w}[1]) \tag{2.19}$$

Si despejamos $\Delta\vec{w}[1]$ en la ecuación 2.19, obtenemos la ecuación 2.20:

$$\Delta\vec{w}[1] = \frac{\sum_{\vec{x} \in R_i} (\vec{x}[0] - \vec{p}[0])^2 * \vec{w}[0]}{\sum_{\vec{x} \in R_i} (\vec{x}[1] - \vec{p}[1])^2} - \vec{w}[1] \tag{2.20}$$

La generalización para K dimensiones se define en la ecuación 2.21. El primer componente ($\vec{x}[0]$) se toma arbitrariamente como referencia (podría haberse tomado

cualquier otro).

$$\Delta\vec{w}[j] = \frac{\sum_{\vec{x}\in R_i}(\vec{x}[0] - \vec{p}[0])^2 * \vec{w}[0]}{\sum_{\vec{x}\in R_i}(\vec{x}[j] - \vec{p}[j])^2} - \vec{w}[j] \tag{2.21}$$

Ya que el primer componente se ha tomado como referencia podemos fijar el peso asociado a este componente a un valor fijo, por ejemplo $\vec{w}[0] = 1$. Así la ecuación previa puede simplificarse a la ecuación 2.22.

$$\Delta\vec{w}[j] = \frac{\sum_{\vec{x}\in R_i}(\vec{x}[0] - \vec{p}[0])^2}{\sum_{\vec{x}\in R_i}(\vec{x}[j] - \vec{p}[j])^2} - \vec{w}[j] \tag{2.22}$$

Ahora se pueden calcular los nuevos pesos de un prototipo utilzando la fórmula 2.23.

En la fórmula se incluye un parámetro $z \in [0, 1]$ para ajustar la intensidad del cambio de los pesos en cada iteración.

$$\vec{w}[j]^{t+1} = \begin{cases} \vec{w}[j]^t, & \text{si j=0} \\ \vec{w}[j]^t + z * \Delta\vec{w}[j], & \text{en otro caso} \end{cases} \tag{2.23}$$

2.4. Aprendizaje Relacional

El aprendizaje relacional se distingue de otras áreas del aprendizaje automático por el tipo de representación del dominio que utiliza. En lugar de utilizar una única relación, como se hace en el aprendizaje proposicional, donde cada observación se describe mediante una tupla, una conjunción de pares atributo-valor, se utiliza un conjunto de relaciones, donde una observación puede ser descrita mediante conjuntos de tuplas en distintas relaciones, equivalente a una representación en lógica de primer orden. Por esta razón los algoritmos de aprendizaje relacional deben afrontar una serie de dificultades que no están presentes en los algoritmos proposicionales.

La principal ventaja de la representación relacional respecto a la proposcional es que se dispone de una mayor expresividad, que permite aprender en dominios con una estructura cuya complejidad haría imposible manejar en una representación proposicional.

Entre las principales aplicaciones de aprendizaje relacional, enumerados en [1], se

encuentran varias en biología y medicina, principalmente en el diseño de fármacos y en la predicción de aparición de enfermedades; en ingeniería mecánica; en procesado de lenguaje natural; y en análisis de información empresarial.

2.4.1. Representación relacional de dominios

En aprendizaje relacional el formato de representación es un factor importante, ya que no es tan trivial como suele serlo en aprendizaje proposicional, donde es suficiente representar cada ejemplo como una tupla individual en una tabla, si no que debe representarse cada instancia en función de las tuplas asociadas a ella en cada relación del dominio.

Existen dos maneras de representación principales. En la primera se describe un dominio como un conjunto de hechos, definidos a partir de una serie de predicados. Esta manera de representar el dominio es la más utilizada en Programación Lógica Inductiva (ILP). Cada instancia relacional se describe implícitamente A continuación se muesta un ejemplo de esta representación:

```
persona(Juan, 22, Cuenca).
coche(Juan, Fiat, M-534534).
coche(Juan, Ford, M-342234).
multa(M-342234, 100).
multa(M-342234, 500).
casa(Juan,Cuenca).

persona(Helena, 32, Toledo).
casa(Helena,Toledo).
casa(Helena,Madrid).
hijo(Helena,Maria).

persona(Maria,8,Toledo).
```

En este ejemplo se representa información sobre un conjunto de personas, sus posesiones (coches y casas) e hijos. Cuando se define un dominio de esta manera se dispone de un conjunto de objetos que se relacionan en cada instanciación de una relación. Para cada relación se definirá el tipo de cada objeto, por ejemplo en la

relación persona se podría fijar que el primer objeto se debe representar mediante una cadena de caracteres, el segundo por un número entero y el tercero por otra cadena. También se establecerán las referencias de un objeto en una relación, por ejemplo, en la relación coche, el primer objeto debe ser una persona, un objeto que debe aparecer en primer lugar en alguna instanciación existente de la relación *persona*.

En este ejemplo cada hecho *persona* se corresponde con una instancia relacional. A partir de ese hecho, sabiendo el nombre de la persona, la descripción de la instancia se completa con los hechos que se refieren a la misma persona, los hechos que se refieran a alguna de los objetos que aparezcan en alguno de esos hechos, y así sucesivamente.

Por ejemplo Juan es un hombre de 22 años que vive en Cuenca, tiene dos coches (uno de ellos con dos multas), y una casa en Cuenca. Esta instancia relacional puede entenderse como un árbol (como se muestra en la figura 2.11).

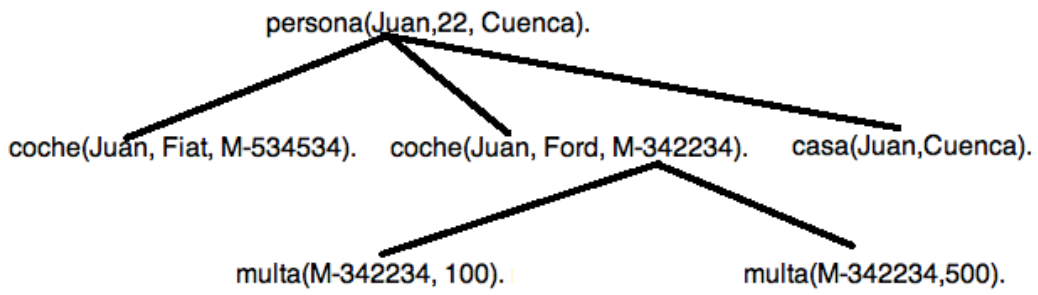


Figura 2.11: Ejemplo de instancia relacional

La segunda aproximación se basa en la definición de un esquema relacional, donde se describe independientemente cada relación y la dependencia entre relaciones se describe mediante el uso de claves ajenas. Una clave ajena es una característica de una relación a la que se impone la restricción de que todos sus valores deben aparecer en una característica de otra relación. A partir de este mecanismo se pueden definir una instancia relacional mediante una tupla en una relación raíz, que se identifica mediante una clave unívoca, y cuya descripción se distribuye en tuplas de varias relaciones relacionadas entre sí mediante claves ajenas. Este tipo de representación es predominante en las bases de datos relacionales.

Si considerasemos el ejemplo anterior representado de esta manera, obtendríamos 5 relaciones, y cada instanciación de la relación constituiría una tupla de la relación.

La correspondencia entre objetos se representan ahora como claves ajenas que asocian tuplas con tuplas de otra relación.

Persona		
Nombre	*Edad*	*Ciudad*
Juan	22	Cuenca
Helena	32	Toledo
María	8	Toledo

Coche		
Propietario	*Marca*	*Matrícula*
Juan	Fiat	M-534534
Juan	Ford	M-342234

Casa	
Propietario	*Ciudad*
Juan	Cuenca
Helena	Toledo
Helena	Madrid

Multa	
Matrícula	*Importe*
M-342234	100
M-342234	500

Hijo	
Padre	*Hijo*
Helena	María

Figura 2.12: Relaciones del esquema de izquierda a derecha, y de arriba a abajo: *Persona, Coche, Casa, Multa, Hijo*

2.4.2. Aproximaciones al aprendizaje automático relacional

Podemos considerar tres aproximaciones principales para resolver problemas de aprendizaje automático que requieran una representación relacional.

La primera manera es intentar evitar la representación relacional y realizar una proposicionalización del dominio, para poder aplicar directamente algoritmos de aprendizaje proposicional. El principal inconveniente de esta aproximación (explicado en [10]) es que existen dominios donde la proposicionalización no es posible (por que requieren la recursión, por ejemplo), o el número de características puede crecer exponencialmente respecto al problema original.

Otra manera de utilizar la representación relacional es aprovechar la expresividad de la lógica de primer orden para construir una teoría que describa un determinado concepto. Esto se conoce como ILP (*Inductive Logic Programming* o Programación Lógica Inductiva). Generalmente el resultado de un algoritmo de ILP consistirá en un conjunto de reglas (que normalmente serán cláusulas de Horn) o un programa lógico (utilizando el lenguaje de programación Prolog). Los algoritmos de inducción de programas lógicos han sido estudiados, principalmente, desde principios de los años noventa y han tenido bastantes aplicaciones relevantes en el campo de la bi-

ología (sobre todo en bioinformática). Los sistemas actuales de ILP más destacables incluyen Aleph [1] y Tilde [18].

La tercera manera de realizar aprendizaje automático en dominios relacionales es adaptar las técnicas conocidas en aprendizaje proposicional, para hacerlas aplicables a dominios relacionales. Así, varios de los algoritmos más conocidos en aprendizaje proposicional tienen una versión relacional.

En este último caso es destacable la facilidad de adaptación que presentan los algoritmos basados en instancias. Ya que estos algoritmos se basan en la asunción de que es posible medir numéricamente el grado de diferencia entre dos ejemplos a partir de una cierta distancia, si conseguimos definir una distancia que admita el caso relacional, podríamos aplicar los algoritmos sin modificaciones importantes.

2.4.3. Aproximaciones relacionales basadas en instancias

El algoritmo de referencia en clasificación supervisada basada en instancias es RIBL (Relational Instance Based Learning), que adapta el algortimo k-NN al caso relacional [1].

Dentro de *clustering* también existen varias aproximaciones basadas en distancias relacionales. Por ejemplo, el algoritmo FORC [1] es una adaptación al caso relacional del algoritmo K-Medias, explicado anteriormente.

La definición de la métrica de distancia utilizada en un algoritmo basado en instancias es vital, porque la precisión de las comparaciones en las que se base el modelo creado por el algoritmo depende de como se mida la similitud o diferencia entre instancias.

Como se comentó anteriormente, una instancia relacional se diferencia de una instancia proposicional, en que se compone de distintas tuplas relacionadas entre sí, siendo el número de éstas variable.

Cuando se define una medida de distancia relacional se debe tener en cuenta un caso especial, la distancia entre claves ajenas distintas. Por esta razón las medidas de distancias relacionales deben ampliar la definición de las distancias proposicionales para definir la distancia entre conjuntos de tuplas a partir de las tuplas individuales que componen estos conjuntos.

[1] http://www.comlab.ox.ac.uk/activities/machinelearning/Aleph/aleph.html

Generalmente, para considerar la distancia entre tuplas individuales se mantiene alguna de las definidas en el punto 2.1.3, y la diferencia entre distancias relacionales se encuentra en la manera en que miden la distancia entre conjuntos de tuplas.

A continuación se describen algunas de las medidas de distancia relacional más relevantes [12]. Para definir formalmente las distancias se consideran dos conjuntos $A = \{a_i | a_i \in A, i \in [1, |A|]\}$ y $B = \{b_j | b_j \in B, j \in [1, |B|]\}$. D define la distancia entre conjuntos y d la distancia entre elementos del conjunto.

Las principales distancias entre conjuntos son las siguientes:

- *Single Linkage*: es igual a la mínima distancia entre cualquier miembro de A con cualquier mienbro de B.

$$D(A, B) = \min_{i,j}(d(a_i, b_j)) \tag{2.24}$$

- *Complete Linkage*: es igual a la máxima distancia entre cualquier miembro de A con cualquier mienbro de B.

$$D(A, B) = \max_{i,j}(d(a_i, b_j)) \tag{2.25}$$

- *Average Linkage*: considera la distancia media entre todos los posibles pares.

$$D(A, B) = \frac{\sum_{i,j}(d(a_i, b_j))}{|A||B|} \tag{2.26}$$

- *Suma de las distancias mínimas*: media de las distancias mínimas entre todos los posibles pares.

$$D(A, B) = \frac{1}{2}(\sum_{a_i} \min_{i,j}(d(a_i, b_j)) + \sum_{b_i} \min_{i,j}(d(b_i, a_j))) \tag{2.27}$$

- *RIBL*: esta es la distancia propuesta en [11].

$$D(A, B) = \begin{cases} \sum_{a_i} min_{b_j}(d(a_i, b_j))/|B| \text{ si } |A| < |B| \\ \sum_{b_i} min_{a_j}(d(a_i, b_j))/|A| \text{ si } |A| \geq |B| \end{cases} \tag{2.28}$$

- *Haussdorf*:

$$D(A,B) = max(\max_{a_i}(\min_{b_j}(d(a_i,b_j), \max_{b_i}(\min_{a_j}(d(b_i,a_j)) \tag{2.29}$$

- *Tanimoto*: esta distancia se utiliza cuando no se dispone de una distancia entre los elementos de los conjuntos.

$$D(A,B) = \frac{|A| + |B| - 2|A \cup B|}{|A| + |B| - |A \cup B|} \tag{2.30}$$

2.5. Relational Nearest Prototype Classifier

RNPC [4] es una extensión de ENPC para dominios relacionales. La principal diferencia respecto a ENPC es que en la fase de movimiento cada prototipo se corresponde con la instancia que minimiza la distancia al resto de la región, es decir, es el medoide de la región, y no con el centroide, que se corresponde con la instancia que se encuentra en el centro de la región. La razón de esta diferencia es que el uso del centroide suele implicar la síntesis de sus características respecto a los valores de otras instancias. En el caso proposicional esta asignación de valores se basa en la media aritmética de los atributos numéricos, y los valores más frecuentes para los atributos nominales, pero en el caso relacional la asignación de valores a las diferentes características de cada relación debe tener en cuenta que en cada relación pueden aparecer distintas tuplas que se refieren a tuplas de una relación distinta, lo que implica además, que este cálculo sea más costoso que en el caso proposicional.

En el algoritmo ENPC, el representante de cada región es el centroide de la región (el punto en el espacio que minimiza la distancia media a los demás puntos de la región). Este punto puede no correspondenderse con ninguna instancia de la región.

El problema en el caso relacional es que cada instancia se describe, no como un vector de valores, sino como conjuntos de tuplas pertenecientes a relaciones distintas. El problema se ilustra con un ejemplo:

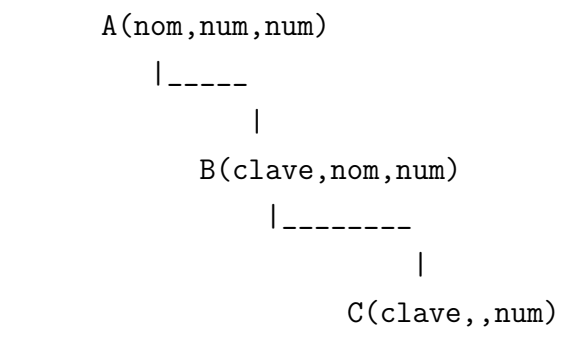
- Si definimos la siguiente región con tres instancias en un dominio con una única relación:

a_1	a_2	a_3	a_4
13	0.2	4	a
22	0.18	7	b
16	0.12	7	a

Podemos definir una instancia representante de esta región como una tupla con la media geométrica (esta media podría ser ponderada si considerásemos los pesos asociados a cada atributo, pero suponemos que se utiliza la distancia euclídea para simplificar) de cada atributo numérico, y el valor mayoritario en caso de que el atributo sea nominal:

$centroide =< 17,0{,}16,6,a >$

- En cambio, si definimos la siguiente región de tres instancias en un dominio con tres relaciones y el siguiente esquema relacional:



y las siguientes instancias (los subíndices distinguen tuplas de una misma relación):

```
A(val1,12,17)
  B(val1,a,12)
  B(val1,b,2)
    C(b,12)

A(val2,13,14)
  B(val2,a,12)

A(val3,6,1)
  B(val3,d,12)
    C(d,12)
```

En este caso deberíamos seleccionar el número de tuplas de cada relación que tendrá el medoide, aparte de asignar valores a cada una de ellas. Por lo tanto en este caso la selección del centroide es problemática.

Aunque se ha estudiado la definición del centroide en dominios relacionales, la solución que suele tomarse en los algoritmos relacionales basados en distancias es la elección del medoide en vez del centroide. El medoide es el punto perteneciente al conjunto de puntos que definen la región que minimiza la distancia al resto. En el ejemplo anterior el medoide se correspondería con alguna de las instancias de la región (la que minimizase la distancia a las otras dos, dependiendo de la medida de distancia elegida).

La diferencia puede observarse gráficamente en la figura 2.13.

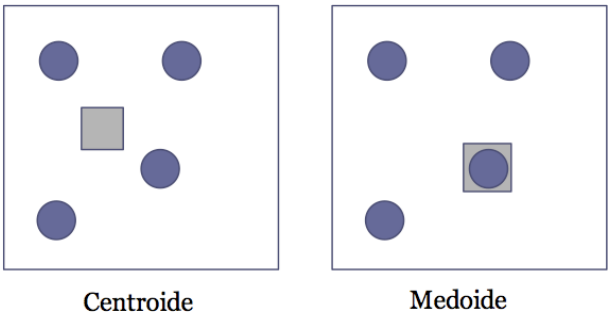


Figura 2.13: □: representante de la región, ◯: instancia de la región

2.6. Weka y Relational Weka

El desarrollo de este proyecto se basa en la plataforma de minería de datos Weka (*Waikato Environment for Knowledge Analysis*, Entorno para Análisis del Conocimiento de la Universidad de Waikato) [17], un conocido software para aprendizaje automático y minería de datos escrito en Java y desarrollado en la Universidad de Waikato. Weka es un software libre distribuido bajo licencia GNU-GPL, que oferta un API para la reutilización del código de la plataforma y la automatización de su uso.

Weka implementa diversos algoritmos de aprendizaje proposicional. Para representar un conjunto de ejemplos utliza el formato ARFF (*Attribute-Relation File Format*, Formato de archivo atributo-relación). En un archivo ARFF se define una

relación mediante la enumeración de sus atributos, y las tuplas que representan los ejemplos. Para cada atributo se define su tipo (numérico, nominal, o cadena de texto), y, en caso de que sea nominal, el conjunto de valores que puede tomar.

A continuación se muestra el ejemplo de dos archivos ARFF diferentes:

```
@relation relacion1
@attribute at1 numeric
@attribute at2 {a,b,c}

@DATA
12,a
15,c
17,b

@relation relacion2
@attribute at1 {a,b,c}
@attribute at2 string
@attribute at3 string

@DATA
a,cad1,12.5
a,cad2,-10.01
b,cad3,14.1
c,cad4,2.2
c,cad5,1.2
c,cad6,-7.2
```

Relational Weka [2] es una extensión de Weka, que incorpora algoritmos de aprendizaje relacional basado en instancias. Esta extensión incorpora diversos algoritmos basados en distancias, e implementa varias medidas basadas en distancias. Para representar conjuntos de ejemplos relaciones utiliza el formato MTARFF.

MTARFF (*Multi-Table Attribute-Relation File Format*, Formato de archivo multitabla atributo-relación) es un formato que permite la representación de esquemas relacionales en dominios de aprendizaje automático. Este formato se define como una extensión del formato ARFF.

[2] http://cui.unige.ch/ woznica/rel_weka

La principal diferencia entre ARFF y MTARFF es que este último permite definir dependencias entre distintas relaciones, mediante el uso de claves ajenas. Una relación que haga referencia a otra, debe contener un atributo cuyos valores se correspondan con los valores de la relación referenciada.

Las relaciones que componen el dominio y las referencias de unas a otras se definen en un archivo con extesión MTARFF, mientras que cada una de las relaciones individuales se define en un archivo ARFF separado.

Por ejemplo, el siguiente aechivo representaría un esquema relacional compuesto por las dos relaciones presentadas anteriormente:

```
relation relacion1
file relacion1.arff
class clase

relation relacion2
file relacion2.arff
foreign:at1:relacion1:at2
```

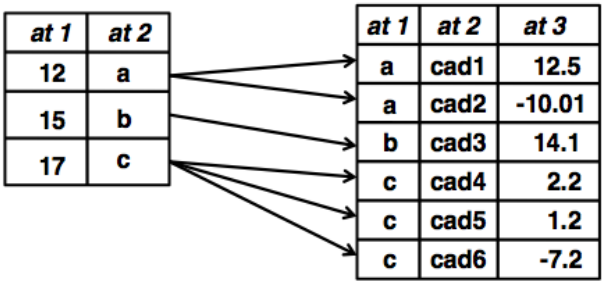


Figura 2.14: Dependencia entre *relacion*1 y *relacion*2

La dependencia se muestra gráficamente en la figura 2.14. Como el atributo $at2$ en la primera relación es clave ajena del atributo $at1$ de la segunda, el valor de este atributo en cada tupla de la segunda relación se corresponde con uno de los valores que aparecen en la primera relación. Así, se puede considerar cada tupla de la primera relación como la raíz de una instancia relacional.

Aunque el formato MTARFF presenta algunas deficiencias (como el que la definición de claves ajenas sólo permita utilizar un atributo y no conjuntos de atributos), permite definir conjuntos de datos relacionales de una manera sencilla.

2.7. Evaluación de resultados

Los resultados obtenidos en los algoritmos de aprendizaje deben evaluarse de alguna manera para conocer cómo es capaz de generalizar sus predicciones el modelo creado, ya que las hipótesis obtenidas podrían ajustarse demasiado a los datos de entrenamiento y realizar predicciones erróneas en ejemplos que no estén presentes en los datos de entrada (este hecho se conoce como sobreadecuación u *overfitting*).

Las principales técnicas de validación para algoritmos de clasificación se basan en la medición de la precisión del modelo obtenido con un conjunto de ejemplos no contenido en los datos de entrenamiento. La manera más sencilla de hacer esto es reservar un porcentaje del total de datos disponibles para la evaluación, entrenando con los ejemplos restantes. Aunque este técnica es sencilla presenta el problema de la selección del porcentaje de evaluación y la posibilidad de que el resultado obtenido tenga un sesgo debido a la manera de seleccionar las instancias para cada conjunto.

Seguramente la técnica más frecuente de evaluación para algoritmos de aprendizaje supervisado es la validación cruzada. Este método de validación consiste en la división del conjunto total de datos en un número de particiones determinado (o *cubetas*). Se realizan tantas iteraciones del algoritmo como número de estas particiones. En cada iteración se utilizará una partición determinada como conjunto de validación y el resto de cubetas como conjunto de entrenamiento. Un ejemplo de este proceso se muestra en la figura 2.15.

Finalmente los resultados del algoritmo se evalúan mediante la media aritmética del éxito de clasificación obtenido en cada cubeta:

$$acierto = \frac{\sum_{cubeta=1}^{k} acierto(cubeta)}{k} \tag{2.31}$$

Ya que el acierto obtenido en los 10 cubetas puede considerarse como una variable aleatoria, ya que cada una de las particiones se ha formado a partir de una selección de ejemplos al azar, es necesario conocer la desviación estándar. Esta medida permite saber cuán representativo es el resultado obtenido en la validación. La desviación

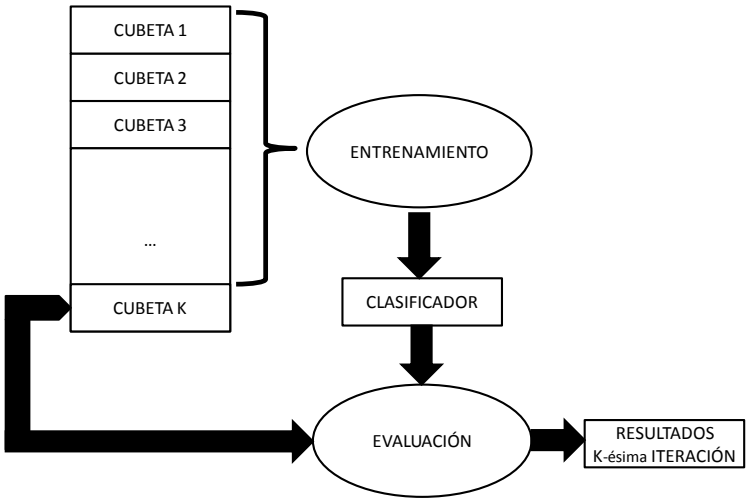


Figura 2.15: Iteración k de una validación cruzada

estándar para k observaciones se define mediante la siguiente ecuación:

$$\sigma = \sqrt{\frac{1}{k} * \sum_{i=1}^{k} (acierto_i - \mu)^2} \tag{2.32}$$

donde μ es la media de todas las observaciones.

Los principales inconvenientes que pueden observarse es que se requiere realizar tantas ejecuciones del algoritmo como número de cubetas, lo cual puede requerir mucho tiempo en algunos casos, y que debe seleccionarse en número de cubetas. Generalmente el número de cubetas utilizado es 10 [8].

Un caso especial de la validación cruzada es el conocido como *Leave one out*. En este método se eligen tantas cubetas como ejemplos disponibles. De esta manera el algoritmo es evaluado con modelos muy similares (ya que sólo eliminamos una instancia del conjunto de entrenamiento). En este caso el tiempo puede ser excesivo, y además el método puede presentar otro tipo de desventajas [14].

También debemos tener en cuenta que en los algoritmos con una componente estocástica, es necesario realizar distintas ejecuciones con una misma configuración de parámetros para poder evaluar los resultados medios en un conjunto de datos particular, puesto que distintas ejecuciones, sobre los mismos datos, y con los mismos parámetros pueden producir resultados distintos.

Capítulo 3

Local Feature Weighting Relational Nearest Prototype Classifier

En este capítulo se presentan y explican los principales componentes del algoritmo Local Feature Weighting Relational Nearest Prototype Classifier (LFW-RNPC). LFW-RNPC es un algoritmo de clasifiación supervisada para dominios relacionales basado en prototipos, que realiza una ponderación local de características automática.

En la sección 3.1 se comentan los distintos pasos de la ejecución del algoritmo, destacando las diferencias respecto a otros algoritmos, descritos en la secciones 2.3 (ENPC) y 2.4 (LFW-NPC), para relacionar los pasos entre sí. En la sección 3.2 se describe formalmente el cálculo del representante de cada de cada región, el medoide, que representa un prototipo de la población. En la sección 3.3 se describe la medida de distancia propuesta para incorporar la ponderación de características a la métrica relacional, explicando descisiones de diseño y modificaciones sobre otras distancias relacionales, y cómo afecta la ponderación de características al cálculo de la distancia. El cálculo de la distorsión y de la actualización de los pesos locales asociados a la medida de distancia explicada en el punto anterior se explica en la sección 3.4. Por último, en la sección 3.5 se enumeran los parámetros de entrada del algoritmo y se comentan sus efectos sobre la ejecución del algoritmo.

3.1. Flujo de ejecución

Básicamente el algoritmo sigue el flujo de ejecución del algoritmo RNPC y ENPC, a lo que se añaden las fases de actualización de pesos. Ya que varias de las fases son similares a las fases del algoritmo ENPC (punto 2.3), sólo se explicarán en detalle aquellas donde se haya producido algún cambio significativo, y se recordará el propósito de algunas de ellas.

En la figura 3.1 se describen los pasos que sigue el algoritmo:

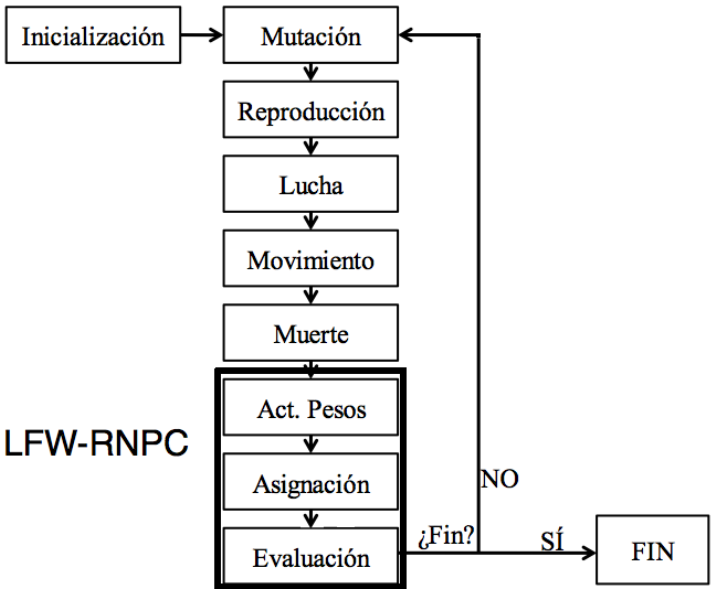


Figura 3.1: Diagrama de flujo de LFW-RNPC

1. *Inicialización*:
   En esta fase se elige el prototipo inicial aleatoriamente entre todas las instancias de entrada. El prototipo representa a la región que comprende a todas las instancias de aprendizaje del algoritmo en una única región, y con unos pesos inicializados a 1.

2. *Mutación*:
   Cada prototipo toma la clase mayoritaria en su región (este paso es igual al explicado en el apartado 2.3).

3. *Reproducción*:
   El objetivo de este operador es la creación de nuevos prototipos, de tal manera

que el prototipo que se reproduce pueda aumentar su calidad (este paso es igual al explicado en la sección 2.3).

4. *Lucha*:
   En esta fase los prototipos se traspasan instancias cooperando (cediendo instancias de clases ajenas) o compitiendo (quitando instancias de la propia clase). Esta fase es similar a la explicada en el punto 2.3, con la variación de que ahora los prototipos sólo deciden luchar con prototipos con menor calidad a la suya.

5. *Movimiento*:
   En este paso se selecciona el representante de la región. El cómputo de este representante se basa en la búsqueda del medoide del conjunto de instancias que forman parte de la región, como se explica en el punto 3.2.

6. *Muerte*:
   El propósito de este operador es eliminar a prototipos inútiles en el clasificador (este paso es igual a la explicada en la sección 2.3).

7. *Actualización de pesos*:
   En esta fase para cada prototipo se calcula la distorsión asociada a cada atributo en cada región, y se actualizan los pesos asociados a todos los atributos de todas las relaciones del esquema como se describe en el punto 3.3.

8. *Asignación de instancias*:
   En esta fase se reasignan las instancias a sus prototipos más cercanos. En caso de que algún prototipo quede sin instancias asociadas a su región se elimina de la población. Además de calcular el prototipo más cercano a cada instancia, en esta fase se calculan los vecinos de cada prototipo. Los vecinos de un prototpo son los segundos prototipos más cercanos a alguna instancia de su región (se recuerda que los vecinos de un prototipo son aquellos con los que un prototipo puede luchar).

9. *Evaluación de la iteración*:
   En este paso se evalúa el clasificador compuesto por la población de prototipos resultante tras la iteración. El clasificador se evalúa respecto a un conjunto de instancias seleccionadas al azar de entre las instancias de entrada del algoritmo. El porcentaje de instancias seleccionadas para esta evaluación es un parámetro del algoritmo. En caso de que el éxito de clasificación en la evaluación de esta iteración sea mejor que el alcanzado en iteraciones anteriores se guarda la

población actual de prototipos. Si el éxito de clasificación es igual que el mejor hasta el momento pero el número de prototipos de la población actual es menor también se registra el clasificador como el mejor hasta el momento.

10. *Condición de finalización*:
    En nuestra implementación se ha elegido el número de iteraciones como criterio de finalización. Se ha establecido este criterio porque su sencillez permite observar mejor el desarrollo de la población de prototipos frente a criterios basados en la estabilidad de la población o la variación de pesos.

    El algoritmo devuelve el mejor clasificador (población de prototipos) registrado en las evaluaciones de las distintas iteraciones del algoritmo.

3.2. Cálculo del medoide

El medoide de un conjunto de puntos es aquel punto del conjunto que minimiza la distancia a todos los demás. En el algoritmo LFW-RNPC el representante de cada región es el medoide del conjunto de puntos. Se eligió utilizar el medoide en lugar del centroide porque no se encontró ninguna manera en la literatura de sintetizar un centroide relacional.

A diferencia del centroide proposicional que puede calcularse en tiempo lineal ($O(n)$) en función del número de dimensiones, calculando una media de la características en caso de que sean numéricas, o contando el valor mayoritario en caso de que sean nominales:

```
Para 1..R hacer
  x = instacia[R]
  Para 1..n hacer
   si x[n] es numero centroide[n] = centroide[n] + x[n]
   si x[n] es nominal contar(x[n])
  Fin Para
Fin Para
Para 1..n hacer
   si x[n] es numero centroide[n] = centroide[n] / R
   si x[n] es nominal maximo(x[n])
Fin Para
```

En LFW-RNPC se ha utilizado un algoritmo de fuerza bruta para calcular el medoide. A pesar de que en la literatura existen métodos que mejoran la eficiencia de este cálculo no se han podido utilizar, porque utilizan algún cálculo de la media, por lo tanto se tendría el mismo problema de síntesis del centroide relacional, o porque se basa en la organización de puntos en un plano eucídeo que no es aplicable al caso relacional.

Para calcular el medoide se calcula la matriz de distancias entre instancias de la región. Como asumimos que la relación de distancia es simétrica la matriz será simétrica también, y se puede calcular sólo la mitad de la matriz y consultar la posición inversa cuando quiere conocerse la distancia entre dos puntos que no ha sido calculada. Además, como la distancia entre una instancia consigo misma debe ser 0 (condición necesaria para ser una métrica) también se sabe que la diagonal de la matriz debe ser 0.

$$\begin{pmatrix} 0 & -1 & 3 \\ -1 & 0 & 4 \\ 3 & 4 & 0 \end{pmatrix}$$

Figura 3.2: Ejemplo de matriz de distancias de la región

Otro motivo importante para calcular la matriz de distancias, es que información de esta fase puede reutilizarse en la fase de la actualización de pesos, donde se necesita conocer la distancia asociada a cada atributo del dominio entre cada instancia del región y el representate.

Para optimizar el cálculo de la matriz se optó por paralelizar el cálculo de esta matriz. La paralelización de este cálculo se explica en el anexo A.

Una vez calculada la matriz se suma cada fila y se busca la instancia que minimiza esta suma. Esa instancia es el medoide de la región.

3.3. Medida de distancia

A continuación se presenta la medida de distancia propuesta en este proyecto. Para explicar la medida de distancia se utilizará un ejemplo, basado en el siguiente esquema relacional:

$rel1$		
a_1	a_2	a_3
12	a	val_1
2	b	val_2

$rel2$	
a_4	a_5
12	val_1
9	val_2
1	val_2
5	val_2

Figura 3.3: Ejemplo de esquema relacional

La relación $rel2$ se enlaza con la primera relación $rel1$ mediante el atributo a_5, que referencia valores del atributo a_3.

Como puede observarse en la relación raíz existen dos tuplas que son referenciadas por las tuplas de la segunda relación. Por lo tanto existen dos instancias relacionales en total.

3.3.1. RIBL

La medida de distancia utilizada en este proyecto es similar a la distancia RIBL. Se ha elegido esta distancia por su relevancia en la literatura y por la superioridad de esta medida de distancia respecto a otras cuando se ha utilizado en RNPC [4]. La distancia RIBL se define de la siguiente manera:

$$D(A,B) = \begin{cases} \frac{\sum_{a_i} min_{b_j}(d(a_i,b_j))}{|B|} \text{ si } |A| < |B| \\ \frac{\sum_{b_i} min_{a_j}(d(a_i,b_j))}{|A|} \text{ si } |A| \geq |B| \end{cases} \tag{3.1}$$

Donde D indica una distancia entre conjuntos, y d una distancia entre elementos del conjunto. En nuestro caso se utilizará d para definir la distancia entre tuplas individuales. Para calcular la distancia entre instancias relacionales se comienza comparando la tuplas correspondientes a cada instancia relacional en la relación raíz (en esta relación sólo habrá una tupla por cada instancia relacional).

La medida d define cómo contribuye cada atributo, según su tipo, a la distancia entre las dos tuplas. A continuación se define la distancia asociada a cada atributo a, d_a, en función del tipo de cada atributo (v_1 y v_2 son valores del atributo) en RIBL:

- Atributo nominal: se utiliza una diferencia simple.

$$d_a(v_1, v_2) = \begin{cases} 1 & \text{si} \quad v_1 \neq v_2 \\ 0 & \text{si} \quad v_1 = v_2 \end{cases} \tag{3.2}$$

- Atributo numérico: se calcula la diferencia al cuadrado entre valores, como en la distancia euclídea, normalizando la diferencia entre el máximo valor del atributo:

$$d_a(v_1, v_2) = (\frac{v_1 - v_2}{max(a)})^2 \tag{3.3}$$

- Clave ajena: en este caso se aplica la diferencia entre conjuntos con las tuplas que utilizan alguno de los valores como clave ajena. La función $s(h, a, v)$ define el conjunto de tuplas de la relación h que referencian al valor v en la clave ajena a. En este caso D es la distancia descrita en la ecuación (3.1).

$$d_a(v_1, v_2) = \frac{\sum_{h \in H} D(s(h, a, v_1), s(h, a, v_2))}{|H|} \tag{3.4}$$

  Hay que destacar que las claves ajenas pueden referirse reflexivamente a una relación, o que una de las relaciones subordinadas vuelvan a referirse a relaciones que ya han aparecido, es decir, definir recursivamente una instancia. Por esto es necesario limitar este paso recursivo mediante un parámetro p, que limita la profundidad del cálculo de la distancia. Cuando se llega al límite de esta profundidad en el cálculo entre dos tuplas la distancia entre claves se calcula como si fuesen atributos nominales, utilizando la ecuación (3.2). También se debe considerar el caso en que en una relación existan tuplas que referencien un valor, pero no existan tuplas que se refieran al otro. En ese caso se asume un distancia 1, al igual que en el caso nominal.

Cuando se calcula la distancia de la clave ajena se da el paso recursivo para calcular la distancia de las claves ajenas, donde se consideran las tuplas asociadas a cada instancia relacional.

Una vez calculada la distancia de todos los atributos, la distancia entre tuplas se combinan de manera similar a la distancia euclídea ($I(a)$ se refiere al valor del

atributo a en la tupla I, siendo A_h el conjunto de los atributos de la relación h):

$$d(I_1, I_2) = \sqrt{\sum_{a \in A_h} d_a(I_1(a), I_2(a))} \tag{3.5}$$

Si se utilizára esta distancia para calcular la distancia entre las instancias del ejemplo se obtendría el siguiente resultado:

1. $d(I_1, I_2) = \sqrt{\sum_{a \in A_h} d_a(I_1(a), I_2(a))} = \sqrt{d_{a_1}(12, 2) + d_{a_2}(a, b) + d_{a_3}(val_1, val_2)}$
   Para calcular la distancia entre las tuplas primero se calculan la diferencia entre los valores de cada atributo de la relación.

2. $d_{a_1}(12, 2) = (\frac{(12-2)}{12})^2 = 25/36$

3. $d_{a_2}(a, b) = 1$

4. En el caso del atributo a_3 hay que calcular la distancia entre las tuplas de las relaciones que referencian ese atributo como clave ajena. Como en este ejemplo sólo existe una relación se cálcula la distancia entre los conjuntos de tuplas que se refieran a cada valor según la ecuación (3.1): $d_{a_3}(val_1, val_2) = D(s(rel2, cl, val_1), s(rel2, cl, val_2)$
   $s(rel2, cl, val_1) = \{(12, val_1)\}$
   $s(rel2, cl, val_2) = \{(9, val_2), (1, val_2), (5, val_2)\}$
   $max\{|s(rel2, cl, val_1)|, |s(rel2, cl, val_2)|\} = |s(rel2, cl, val_2)| = 3$
   $D(s(rel2, cl, val_1), s(rel2, cl, val_2) = min\{d_{a_4}(12, 9), d_{a_4}(12, 1), d_{a_4}(12, 5)\}/3$
   $= min\{(\frac{12-9}{12})^2, (\frac{12-1}{12})^2, (\frac{12-4}{12})^2\}/3 = 1/24$

5. El último paso consiste en combinar los valores asociados a todos los atributos:
   $d(I_1, I_2) = \sqrt{25/36 + 1 + 1/24} = \sqrt{125/72} = 1{,}3176$

3.3.2. Distancia LFW-RNPC

Para incorporar la ponderación de características a la distancia, se deben considerar los pesos asociados a los atributos de las relaciones que componen el esquema relacional. Ya que estos pesos son locales cada prototipo de la población tendrá su conjunto de pesos propio: $W_p = \{\vec{w}_{p,h} \in \Re^{|A_h|} | h \in H\}$.

Si se aplicase directamente esta ponderación a la distancia RIBL sólo tendría que multiplicarse el peso asociado a cada atributo en el cálculo de distancia entre tuplas.

$$d(I_1, I_2) = \sqrt{\sum_{a \in A_h} w_{p,a} * d_a(I_1(a), I_2(a))} \tag{3.6}$$

Sin embargo se ha decidido cambiar la medida RIBL en varios aspectos para incorporar la ponderación:

1. *Normalización de atributos numéricos*
   Para realizar una comparación de valores numéricos relativa al rango de cada atributo RIBL normaliza estos valores en la comparación. La normalización se realiza dividiendo la diferencia de los valores entre el máximo valor posible del atributo presenta conflictos respecto al peso asignado al atributo en el cálculo de la distorsión de la región. Ya que el peso asociado al atributo ponderará el valor de la diferencia se ha omitido la normalización de la diferencia de atributos numéricos.

2. *Elevación al cuadrado de la diferencia numérica entre atributos*
   En la distancia presentada anteriormente la diferencia entre los valores de cada atributo es elevada al cuadrado. Si se obvia la normalización en el cálculo de la diferencia entre valores numéricos, como señalamos en el punto 1, esto puede implicar un problema en el cálculo de la distancia, ya que si la diferencia entre valores es muy grade, el peso asociado al atributo es irrelevante en el cálculo de la distancia. A continuación se muestra un ejemplo:

   - Si se tienen los siguientes pesos

     $$W = \{(10^{-10}, 10)\}$$

   - Comparando las siguientes instancias, que pertencen a un esquema relacional con una sola relación de dos atributos numéricos:

a_1	a_2
10^{10}	2
0	2

- Y se calcula la distancia ponderada:

$$d(I_1, I_2) = \sqrt{(10^{10} - 0)^2 * 10^{-10} + (2 - 2) * 10} =$$

$$\sqrt{(10^{10})^2 * 10^{-10}} = \sqrt{10^{20} * 10^{-10}} = 10^5$$

- Un atributo muy poco significativo para la comparación entre distancias ha provocado que haya una diferencia entre dos instancias que tienen el mismo valor en un atributo relevante. Sin embargo si se utilizan diferencias absolutas:

$$d(I_1, I_2) = |(10^{10} - 0) * 10^{-10}| + |(2 - 2) * 10| =$$

$$|10^{10} * 10^{-10}| = 1$$

  El anterior es un resultado más razonable si se interpretan los pesos como medidas de normalización, o importancia de un atributo en la similitud entre las instancias de una misma región.

3. *Pesos asociados a claves ajenas*
   El aporte a la distancia total entre dos instancias por estos atributos al final se resuelve en los atributos de la relación a la que referencia que sí tienen valor, y ya están ponderados. Por lo tanto asociar un peso a una clave ajena ponderaría lo ya ponderado. Por esta razón se decidió mantener el peso asignado a las claves ajenas constante e igual a 1.0.

A continuación se define la función de distancia propuesta en el proyecto.

Se mantienen las diferencias para atributos nominales y claves ajenas de RIBL, pero en los atributos numéricos se calcula la diferencia absoluta entre valores:

$$d_a(v_1, v_2) = |v_1 - v_2| \tag{3.7}$$

En la distancia entre tuplas ahora se incorporan los pesos asociados a los atributos:

$$d(I_1, I_2) = \frac{\sum_{a \in A_h} w_a * d_a(I_1(a), I_2(a))}{|A_h|} \tag{3.8}$$

Ahora se calcula la distancia para el ejemplo del principio de esta sección con

la métrica propuesta, suponiendo que el prototipo de la región tenga los siguientes pesos asociados:

$$W_p = \{w_{rel_1}, w_{rel_2}\}$$

$$w_{rel_1} = (0{,}3, 0{,}4, 1{,}0)$$

$$w_{rel_2} = (5{,}0, 1{,}0)$$

1. Ahora la fórmula para combinar los valores asociados a los atributos de la relación, omitiendo la raíz cuadrada y considerando los pesos asociados a los atributos:

$$d(I_1, I_2) = \frac{\sum_{a \in A_h} w_{h,a} * d_a(I_1(a), I_2(a))}{|A_h|} =$$

$$\frac{w_{h,a_1} * d_{a_1}(12, 2) + w_{h,a_2} * d_{a_2}(a, b) + w_{h,a_3} * d_{a_3}(val_1, val_2)}{3}$$

2. En el atributo numérico ahora se considera la diferencia absoluta:
   $d_{a_1}(a, b) = |12 - 2| \ = 10$

3. $d_{a_2}(a, b) = 1$

4. La distancia entre tuplas es igual, pero al considerar las distancias entre tuplas se considera el peso de cada atributo, al igual que en la relación raíz:
   $s(rel2, cl, val_1) = \{(12, val_1)\}$
   $s(rel2, cl, val_2) = \{(9, val_2), (1, val_2), (5, val_2)\}$
   $max\{|s(rel2, cl, val_1 d)|, |s(rel2, cl, val_2)|\} = |s(rel2, cl, val_2)| = 3$
   $d_{a_3}(val_1, val_2) = D(s(rel2, cl, val_1), s(rel2, cl, val_2) =$
   $min\{w_{a_4} * d_{a_4}(12, 9), w_{a_4} * d_{a_4}(12, 1), w_{a_4} * d_{a_4}(12, 5)\}/3$
   $= min\{5{,}0 * |12 - 9|, 5{,}0 * |12 - 1|, 5{,}0 * |12 - 4|\}/3 = 5{,}0$

5. $d(I_1, I_2) = \frac{0{,}3*10+0{,}4*1+1{,}0*5{,}0}{3} = 19$

3.4. Cálculo de distorsión y actualización de pesos

En esta sección se explica cómo se calculan los pesos aplicados en la medida de distancia definida en la sección anterior. Este cálculo se basa en la adaptación del cálculo de la distorsión explicado en el punto 2.3.2 al caso relacional.

Como se explicó anteriormente la distorsión de un atributo en una región representa la desviación media del valor de ese atributo de las instancias de la región respecto al valor de su representante.

Tras incorporar la ponderación del atributo, la distorsión asociada a éste es:

$$D_a = \frac{1}{||R_i||} \sum_{I \in R_i} d_a(I, p_i) w_{p_i,a} \tag{3.9}$$

Donde $||R_i||$ es el número de instancias en la región del prototipo i, p_i, que tiene asociado un conjunto de pesos W_{p_i}, donde $w_{p_i,a}$ indica el peso de este prototipo asociado al atributo a.

El objetivo de la ponderación es la equiparación de la distorsión de todos los atributos que contribuyan a la distancia. En este caso se ha introducido una modificación respecto al cálculo de la distorsión de LFW-NPC, donde se igualan las distorsiones asociadas a todos los atributos a la distorsión del primero de ellos. Ahora se reformula esta ecualización, igualando las distorsiones al valor 1, para todos los atributos ($a \in A_h$) de todas las relaciones del esquema ($h \in H$):

$$\forall h \in H, \forall a \in A_h, D_a = 1 \tag{3.10}$$

A partir de esta igualdad podemos calcular el valor del peso que debe asociarse a una atributo para igualar su distorsión a 1:

$$D_a = \frac{1}{||R_i||} \sum_{I \in R_i} d_a(I, p_i) * w_{p_i,a} \tag{3.11}$$

$$\frac{D_a * ||R_i||}{\sum_{I \in R_i} d_a(I, p_i)} = w_{pi,a} \tag{3.12}$$

El cálculo del valor del peso se deriva de la sustitución de la ecuación (3.10) en

la ecuación (3.12):

$$w_{p_i,a} = \frac{||R_i||}{\sum_{I \in R_i} d(I, p_i)} \tag{3.13}$$

Ya que la evolución de la población de prototipos causa que las regiones varíen, debido al traspaso de instancias entre unas y otras en la aplicación de los operadores del algoritmo, se calcula el diferencial del peso para actualizar su valor y mantener la ecualización de la distorsión en cada iteración:

$$\Delta w_{p_i,a} = \frac{||R_i||}{\sum_{I \in R_i} d_a(I, p_i)} - w_{p_i,a} \tag{3.14}$$

En la actualización del valor del peso se incorpora un parámetro $z \in [0, 1]$, que regula el grado de actualización entre iteraciones, lo que proporciona la fórmula de actualización de pesos utilizada en LFW-RNPC, como se muestra en la figura 3.1:

$$w_{p_i,a}^{t+1} = w_{p_i,a}^{t} + z * \Delta w_{p_i,a}^{t+1} \tag{3.15}$$

3.5. Parámetros

Se detallan los parámetros de entrada del algoritmo, y cómo afectan a su ejecución.

- *Número de iteraciones*: limita el tiempo de ejecución del algoritmo.
- *Profundidad*: indica el nivel de profundidad máximo que limita el número de pasos recursivos que deben realizarse en el cálculo de distancia entre dos instancias relacionales.
- *Factor de actualización (z)*: indica cuánto debe actualizarse el peso de cada atributo respecto a la variación de la distorsión en cada iteración. En caso de que este parámetro sea igual a 0 el algoritmo se comportará de la misma manera que el algoritmo RNPC. Si es igual a 1 entonces la configuración de pesos de cada región dependenderá únicamente de la distorsión del conjunto de instancias en la iteración actual.
- *Porcentaje de test*: este valor indica cuantas instancias deben reservarse para evaluar el clasificador obtenido en cada iteración. Estas instancias son seleccionadas al azar de entre las instancias de entrada. Al final de cada iteración

se mide el éxito de clasificación en este porcentaje de instancias. Esta medida sirve para seleccionar el clasificador resultante al final del algortimo.

Capítulo 4

Experimentación

En este capítulo se exponen la metodología y resultados obtenidos en la experimentación con el algoritmo LFW-RNPC. En la sección 4.1 se explica cómo se ha realizado la evaluación, y en la 4.2 la configuración de los experimentos. En la sección 4.3 se describen los resultados experimentales obtenidos en dominios artificiales, creados expresamente para comprobar la eficacia del algoritmo. En la sección 4.4 se muestran los resultados obtenidos en dominios relacionales existentes. En la sección 4.5 se muestran resultados en dominios de planificación, utilizados para observar los resultados obtenidos en dominios con una estructura distinta a los de secciones anteriores. Por último, en la sección 4.6 se hace un resumen de los resultados.

4.1. Método de evaluación

El algoritmo se ha evaluado mediante validación cruzada de 10 cubetas. Debido a la componente estocástica del algoritmo es necesario ejecutarlo varias veces ya que cada ejecución tendrá unos resultados distintos. En este caso se decide realizar 4 validaciones cruzadas para cada valor de z. Los resultados que pueden observarse en las tablas de las siguientes secciones se corresponden con las medias aritméticas de las 4 ejecuciones. El resultado de cada iteración de la validación cruzada se obtiene mediante la evaluación del clasificador resultante obtenido tras la ejecución del algoritmo sobre los ejemplos reservados para validación (los correspondientes a cada cubeta).

Las pruebas se han realizado en un ordenador MacBook, con un procesador Intel Core 2 Duo a 2.4 GHz y 2 GB de RAM, DDR2 a 667 MHz.

4.2. Configuración de los experimentos

Se comparan los resultados del algoritmo para distintos valores del parámetro z, el parámetro que determina cuanto se actualizan los pesos asociados a cada atributo en función de la distorsión asociada. Para $z = 0$ el algoritmo es equivalente al algoritmo sin actualización de pesos (RNPC). Los valores elegidos para las pruebas se corresponden a los elegidos en [2] para la experimentación con el algoritmo LFW-NPC. En todos los casos se ha establecido un límite de 200 iteraciones por ejecución.

El porcentaje de test seleccionado para la elección del clasificador que se devuelve como salida del algoritmo es del 20 %. Este parámetro se ha mantenido para igualar la configuración del algoritmo de RNPC, que mantiene este mismo valor en los resultados expuestos en [4].

En cada tabla de resultados se muestran las siguientes columnas (entre paréntesis se muestra la variación estándar de cada valor):

- Entrenamiento(*Entr*): En este caso se muestra el porcentaje de instancias de entrenamiento correctamente clasificadas por el clasificador final. Ya que el criterio de selección del clasificador final no es el porcentaje de precisión de entrenamiento en las instancias de entrenamiento.
- Test: En esta columna se muestra el porcentaje de éxito obtenido por el clasificador final en el conjunto de test. Como se ha comentado anteriormente, este conjunto se selecciona aleatoriamente de entre el conjunto de instancias de entrada del algoritmo.
- Validación(*Val*): En esta columna se muestra el porcentaje de éxito obtenido en el conjunto de validación.
- Número de prototipos(*Prots*): Aquí se muestra el número de prototipos que compone el clasificador de salida.
- Tiempo: En esta columna se muestra el tiempo en milisegundos de cada ejecución de una ejecución del algoritmo (ya que se ha utilizado evaluación cruzada, este timepo se corresponde con una iteración de la evaluación).

Además de la tabla con los resultados obtenidos en cada dominio, se muestran 3 gráficas para mostrar la diferencia en la evolución de la ejecución del algoritmo para distintos valores de z.

4.3. Dominios artificiales

En primer lugar se decidió experimentar con dominios artificiales creados a propósito para comprobar el efecto de la ponderación de características. Además la experimentación en estos dominios tiene la función de validación y verificación del algoritmo, ya que estos dominios no son tan complejos como los dominios experimentales que se utilizan en la sección siguiente.

4.3.1. Dominio 1

Este dominio se crea a propósito para comprobar el efecto de la actualización de pesos en la clasificación. Se intentó simplificar esta relación lo máximo posible, por eso sólo se definieron 2 atributos relavantes para la clasificación, cuya importancia y rango de valores se alternan para cada región:

- *id*: Un identificador de cada instancia.
- *a1*: Un atributo nominal con dos valores posibles: a y b.
- *num1*: Un atributo numérico. Este atributo toma valores entre 1 y 1000.
- *clase*: La clase del atributo. Existen dos clases distintas (0 y 1).

El algoritmo sólo toma en cuenta los atributos *a1* y *num1*, ya que *id* se define como identificador en la definición del esquema.

Se ha generado un conjunto de datos donde para cada instancia el valor a del atributo nominal $a1$ se corresponde con la clase 0 y el valor b con la clase 1. Los valores del atributo numérico se han generado aleatoriamente entre 1 y 1000. Por lo tanto esperamos que en la actualización de pesos, se asigne un peso pequeño al atributo *num1*, ya que al estar sus valores distribuidos aleatoriamente debería tener una distorsión alta en cada región, mientras que el atributo *a1* debería tener un valor mayor.

A continuación se muestran ejemplos de la evolución de una ejecución del algoritmo LFW-RNPC en función del número de prototipos y de la calidad de entrenamiento y test obtenida en cada iteración (figuras 4.1, 4.2 y 4.3).

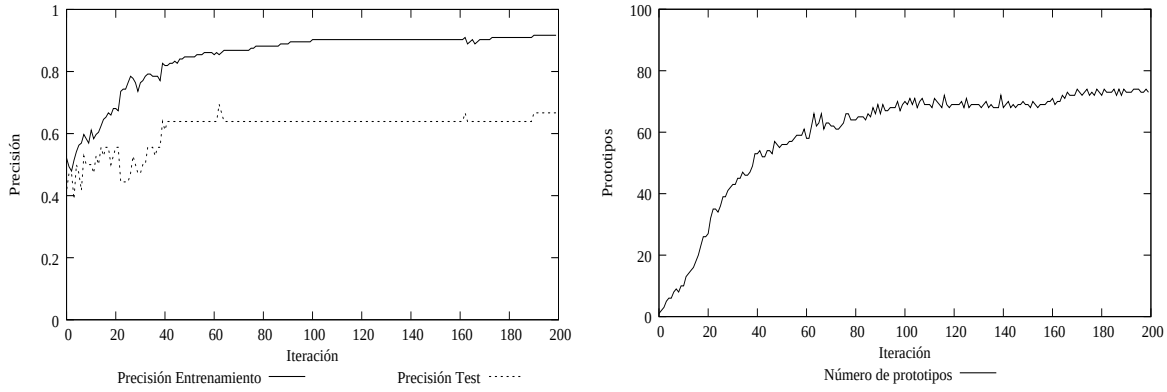


Figura 4.1: Evolución de precisión y número de prototipos con $z = 0{,}0$ en Dominio 1

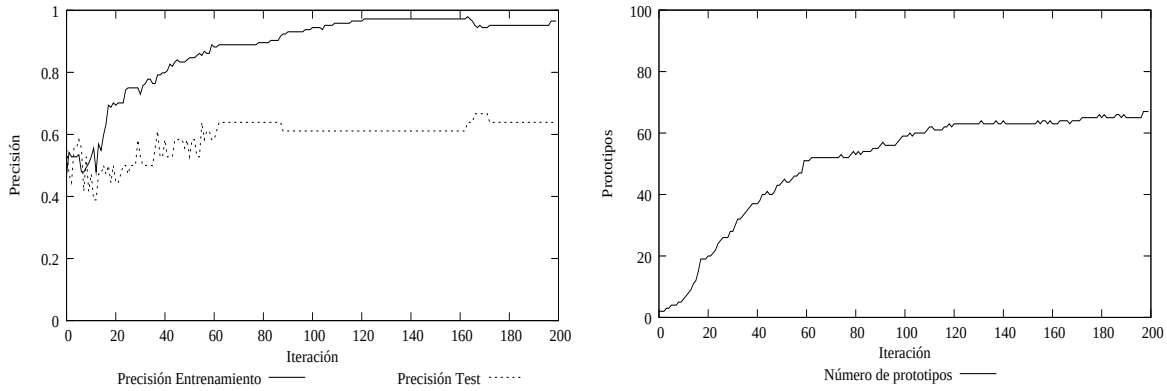


Figura 4.2: Evolución de precisión y número de prototipos con $z = 0{,}5$ en Dominio 1

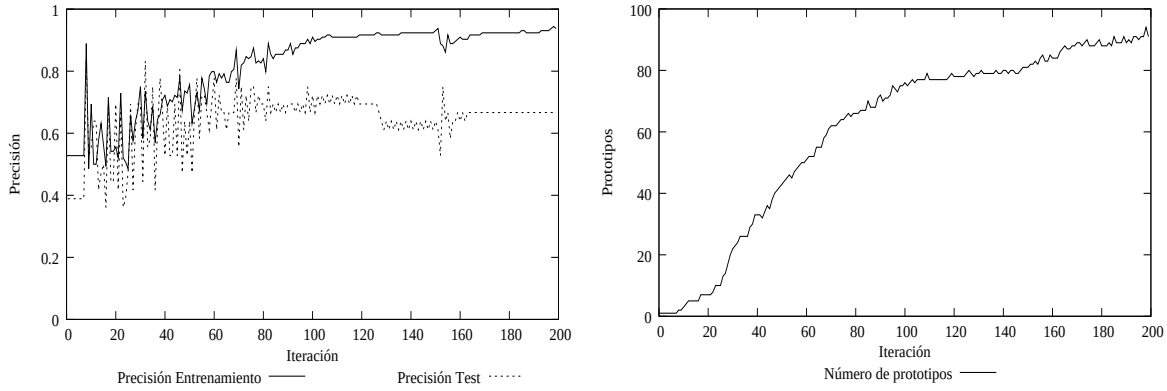


Figura 4.3: Evolución de precisión y número de prototipos con $z = 1{,}0$ en Dominio 1

z	Entr (σ)	Test (σ)	Val (σ)	Prots (σ)	Tiempo (σ)
0,00	75,56 (3,72)	68,75 (0,97)	53,63 (1,13)	41,18 (5,74)	2084,40 (143,25)
0,10	69,65 (2,29)	65,14 (0,14)	51,50 (3,50)	27,45 (1,85)	3957,05 (387,75)
0,30	72,50 (2,15)	66,94 (0,28)	53,00 (0,50)	29,10 (1,50)	4241,30 (983,80)
0,50	78,61 (2,15)	66,39 (0,00)	57,75 (2,25)	39,45 (2,15)	4043,30 (881,00)
0,70	72,15 (1,87)	65,00 (1,67)	56,00 (1,00)	28,60 (3,10)	4065,00 (1073,20)
0,90	77,60 (0,31)	68,33 (0,28)	56,50 (1,00)	39,00 (0,50)	3401,15 (423,55)
1,00	83,96 (2,92)	83,89 (0,28)	75,00 (4,00)	22,10 (3,40)	3350,65 (336,15)

Tabla 4.1: Resultados en Dominio 1

En la tabla 4.1 puede observarse una mejora importante para el caso $z = 1{,}0$. Para este valor de z pueden encontrarse configuraciones de prototipos que optimizan el número de prototipos respecto al porcentaje de éxito en test, aunque no se da una convergencia hacia estas poblaciones, porque la configuración de pesos cambia completamente en cada iteración, porque solo se toma en cuenta la distorsión de la población actual en la actualización de pesos. Los resultados tan bajos en el éxito en validación (en torno al 55 %) pueden darse principalmente por ruido causado por el atributo $num1$, ya que es posible que, al ser sus valores generados aleatoriamente, se hayan generado rangos numéricos asociados a una clase, ofuscando el modelo más simple.

Examinando los resultados puede obervarse que en varios casos el algoritmo es capaz de obtener un clasificador con dos prototipos, por ejemplo en una de las iteraciones con $z = 1$ (y en varias iteraciones con valores altos se han obtenido prototipos similares):

- Prototipo 0:
  - Medoide: $i1, a, 161, 0$
  - Peso en el atributo $a1$: 2.03
  - Peso en el atributo $num1$: 0.0026
  - Clase del prototipo: 0
- Prototipo 1:
  - Medoide: $i128, b, 631, 1$
  - Peso en el atributo $a1$: 1.0

- Peso en el atributo $num1$: 0.004
- Clase del prototipo: 1

4.3.2. Dominio 2

El propósito de este dominio es probar la actualización de pesos en el caso de las claves externas. El esquema de este dominio se compone de 2 relaciones:

- *relacion1*: Esta es la relación raíz, y contiene 200 instancias. En esta relación existen tres atributos:
  - *id*: el identificador de cada instancia relacional.
  - $a1$: es una tributo nominal, que puede tener uno de los siguientes valores: a, b, c, d, e, f, g.
  - *clase*: la clase de la instancia, de entre dos valores posibles (0 y 1).
- *relacion2*: Esta relación tiene 297 tuplas y contiene 2 atributos:
  - $id2$: referencia a la clave ajena *id* de la relación 1. Por cada tupla de relación 1 existen entre 1 y 2 tuplas de esta relación que la referencian.
  - $num1$: es un valor numérico entre 0 y 10.000.

En este dominio se definen tres regiones distintas. El objetivo de estas regiones es comprobar si se puede obtener una configuración de pesos que consiga determinar la importancia de atributos en distintas relaciones del esquema, y para comprobar la ponderación de atributos cuando se debe utilizar una distancia entre conjuntos:

- La primera región se corresponde con la clase 0. En esta región todas las instancias tienen un valor aleatorio en el atributo $a1$ de $relacion1$ raíz y un valor entre 0 y 4 en el atributo $num1$ de $relacion2$.
- La segunda región se corresponde también con la clase 0. En esta región todas las instancias tienen un valor aleatorio en el atributo $a1$ de la relación raíz y un valor cercano a 1000 en el atributo $num1$ en la relación 2.
- La tercera región se corresponde con la clase 1. En esta región todos el valor del atributo $a1$ es siempre b y el valor de las tuplas de la relación 2 es un valor aleatorio entre 1 y 10000.

En este dominio la solución óptima sería un clasificador que constase de 3 prototipos. En las figuras 4.4, 4.5 y 4.6 se muestran ejemplos de evolución del algoritmo en este dominio.

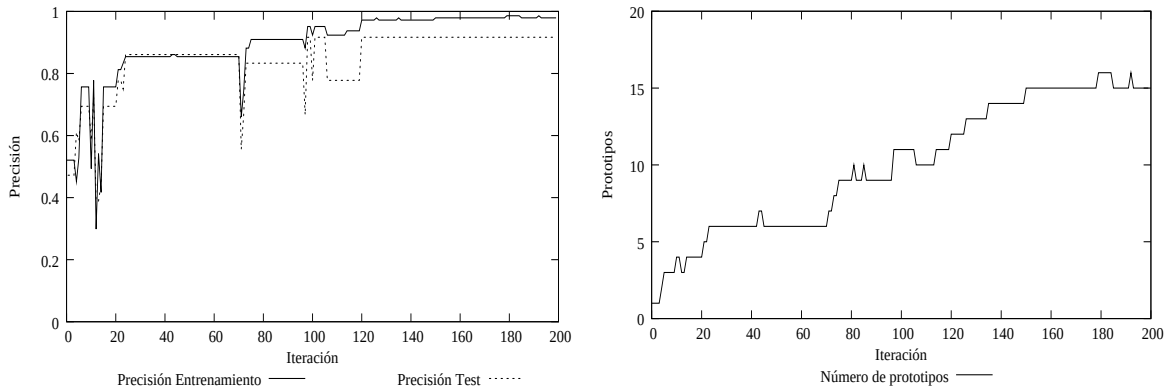


Figura 4.4: Evolución de precisión y número de prototipos con $z = 0{,}0$ en Dominio 2

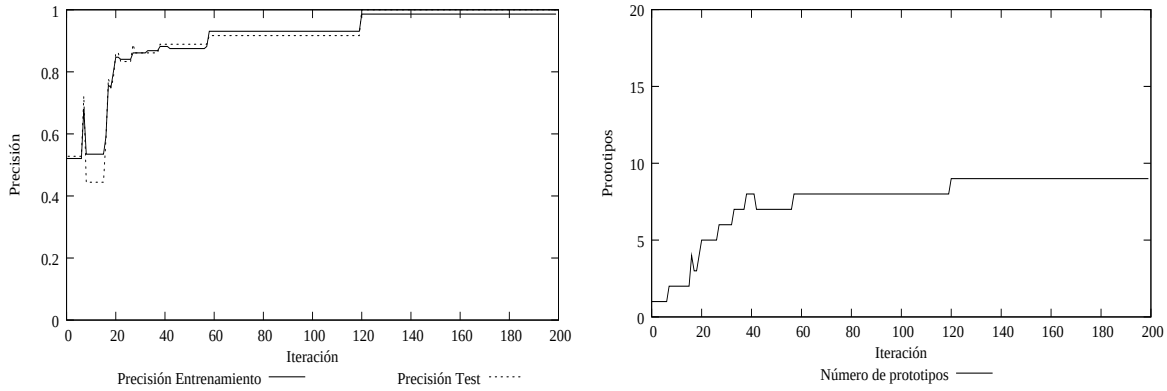


Figura 4.5: Evolución de precisión y número de prototipos con $z = 0{,}5$ en Dominio 2

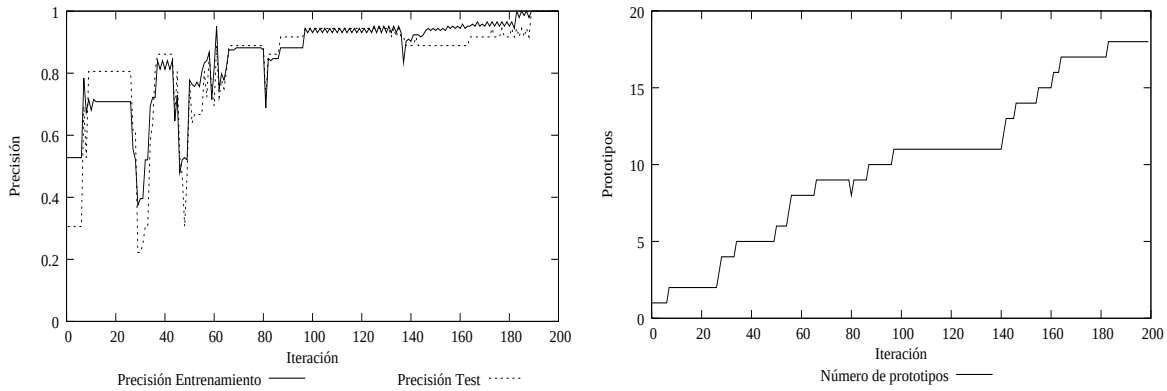


Figura 4.6: Evolución de precisión y número de prototipos con $z = 1{,}0$ en Dominio 2

z	Entr (σ)	Test (σ)	Val (σ)	Prots (σ)	Tiempo (σ)
0,00	91,50 (0,45)	90,43 (0,49)	89,83 (0,56)	10,40 (0,53)	7509,60 (291,80)
0,10	93,47 (0,83)	96,02 (1,36)	92,50 (0,33)	7,50 (0,87)	6901,20 (402,47)
0,30	94,10 (0,60)	97,04 (0,80)	92,67 (1,11)	7,10 (0,60)	7553,43 (652,78)
0,50	94,77 (0,49)	96,20 (1,05)	92,33 (1,22)	9,53 (0,98)	6798,97 (364,11)
0,70	93,52 (0,90)	96,39 (0,56)	94,50 (0,33)	8,33 (0,89)	6679,50 (170,13)
0,90	93,26 (0,46)	96,20 (0,99)	92,67 (1,11)	6,93 (0,98)	6973,47 (182,58)
1,00	91,30 (1,88)	94,10 (1,42)	88,25 (2,50)	7,10 (1,90)	5765,12 (2002,19)

Tabla 4.2: Resultados en Dominio 2

En la tabla 4.2 puede observarse como mejoran los resultados y el número de prototipos disminuye en cuanto se activa la ponderación de pesos (excepto cuando $z = 1{,}0$). Cuando $z = 1{,}0$ aumenta la desviación asociada a las distintas mediciones, obteniendo unos resultados similares a la configuración sin pesos.

En algunas ejecuciones con la ponderación de pesos se obtiene una configuración de 3 prototipos correspondientes a las 3 regiones:

- Prototipo 0
  - Medoide
    $relacion1$: i45,b,0
    $relacion2$: i45,j66,2
  - Clase del prototipo : 0
  - Pesos
    $relacion1$
    a1 1.1288951447895283
    $relacion2$
    num1 1.2482283335757798
- Prototipo 1
  - Medoide
    $relacion1$: i56,f,0
    $relacion2$: i56,j85,1002
  - Clase 0
  - Pesos
    $relacion1$: a1 1.288702246451345

*relacion*2: num1 6.056202443911426

- Prototipo 2
  - Medoide
    *relacion*1: i73,b,0
    *relacion*2: i73,j109,1003
  - Clase 1
  - Pesos
    *relacion*1: a1 4.370485065680423
    *relacion*2: num1 0.038427793723616155

4.3.3. Dominio 3

El propósito de este dominio fue probar la ponderación de pesos con un número de regiones mayor que en dominios anteriores, y un número de tuplas mayor. El dominio se compone de 2 relaciones:

- *relacion 1*: La primera relación es la relación raíz. Esta relación contiene tres atributos:
  - *id*: Un identificador de cada instancia relacional.
  - *color*: Un atributo discreto que puede tener 4 valores: *rojo*, *azul*, *verde* o *amarillo*.
  - *clase*: La clase de la instancia relacional, que puede tomar 2 valores: *a* o *b*.
- *relacion 2*: Es una relación subordinada a la relación 1. Cada instancia de la relación 1 tiene entre 3 y 4 tuplas relacionadas en esta relación.
  - *id*: el identificador que hace referencia a una instancia de la relación 1.
  - *num*: este es un atributo numérico que toma valores entre 0 y 100.

En este dominio definimos 8 regiones distintas en función de cada cambinación de color y valor numéricos de las tuplas asociadas en la relación 2, por lo tanto la

configuración óptima de protipos debería tener 8 prototipos. En las figuras 4.7, 4.9 y 4.8 se muestra el cambio en la evolución de la población respecto al parámetro z.

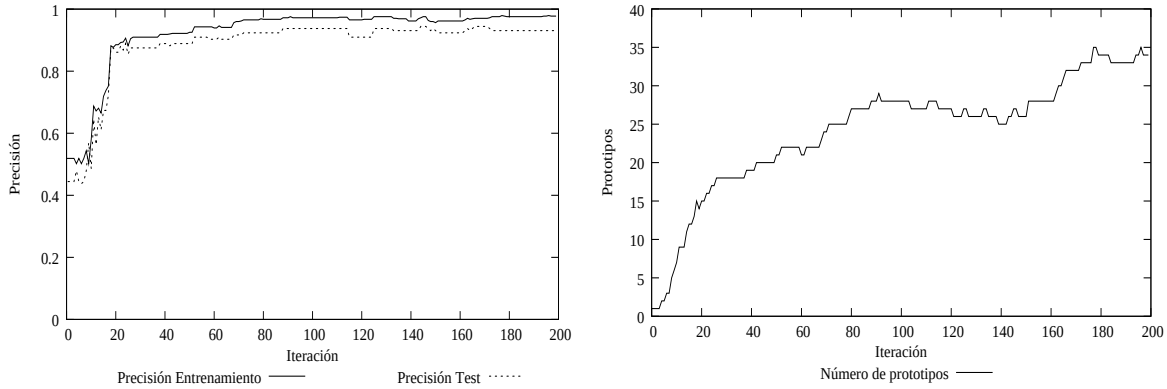


Figura 4.7: Evolución de precisión y número de prototipos con $z = 0{,}0$ en Dominio 3

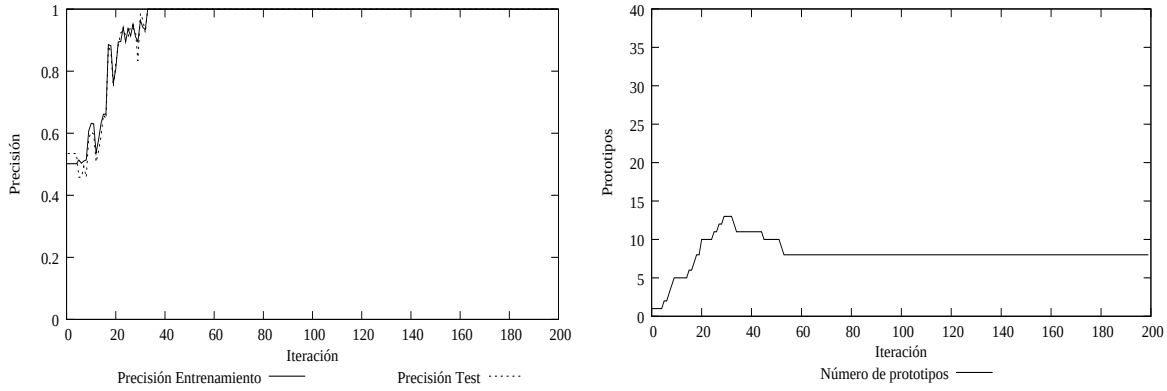


Figura 4.8: Evolución de precisión y número de prototipos con $z = 0{,}5$ en Dominio 3

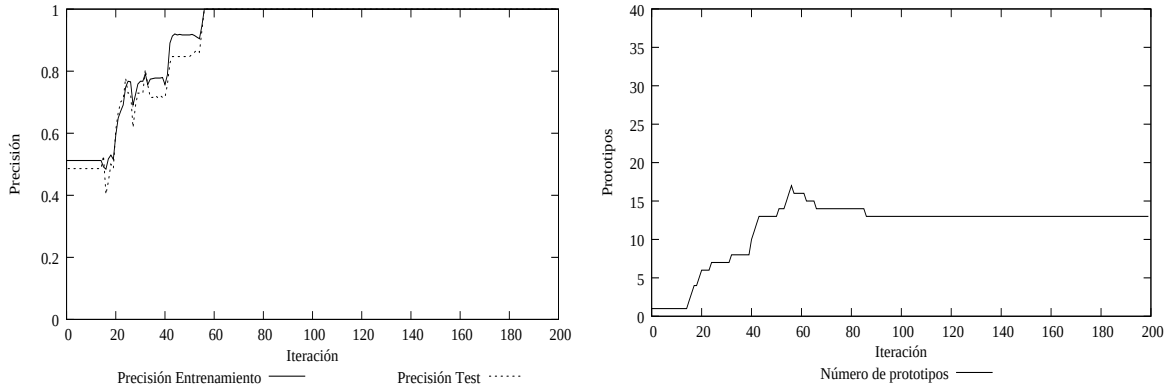


Figura 4.9: Evolución de precisión y número de prototipos con $z = 1{,}0$ en Dominio 3

z	Entr (σ)	Test (σ)	Val (σ)	Prots (σ)	Tiempo (σ)
0,00	96,66 (0,53)	95,54 (0,56)	93,06 (1,19)	33,12 (2,54)	48139,55 (21240,45)
0,10	99,14 (0,13)	99,61 (0,15)	98,54 (0,61)	10,20 (0,33)	71411,00 (31665,00)
0,30	99,13 (0,19)	99,19 (0,34)	97,92 (0,78)	8,60 (0,27)	84412,20 (20204,93)
0,50	99,13 (0,03)	99,21 (0,25)	97,58 (1,22)	9,03 (0,18)	81843,73 (19461,56)
0,70	99,20 (0,14)	99,19 (0,12)	98,29 (1,36)	8,97 (0,49)	84419,97 (19555,69)
0,90	99,54 (0,11)	99,83 (0,03)	99,44 (0,19)	9,15 (0,15)	99034,80 (14784,20)
1,00	93,15 (6,87)	93,40 (6,06)	92,53 (6,72)	14,42 (2,98)	50218,58 (23685,61)

Tabla 4.3: Resultados en Dominio 3

En la tabla 4.3 podemos observar los resultados en este dominio. En los casos donde se activa la ponderación de pesos la media del tamaño de la población se acerca a la configuración óptima, mientras que en la primera configuración el tamaño de la población es mucho más grande, y sus resultados son peores. También se puede observar el efecto de considerar sólo la distorsión actual para la actualización de los pesos de la región ($z = 1{,}0$), donde los resultados obtenidos en media son peores que los obtenidos sin la ponderación, aunque siga disminuyéndose el número de prototipos.

En varios casos en los resultados con la ponderación de pesos se consiguen configuraciones de 8 prototipos que se corresponden con las regiones del dominio.

4.4. Dominios Experimentales

Para probar la efectividad real del algoritmo ahora se utilizan dominios de problemas más complejos, utilizados para probar distintos algoritmos relacionales en la literatura.

4.4.1. Musk

El objetivo en este dominio es predecir la fuerza de moléculas Musk (*Muscle Specific Kinase*) sintéticas. Los ejemplos de este dominio se dividen en dos clases (musk o no musk), y contiene 47 moleculas musk, y 45 que no lo son. Cada ejemplo se describe como un conjunto de tuplas, por lo tanto en la relación raíz cada tupla define la clase de cada ejemplo y su clave externa, mientras que en la relación subordinada se encuentran los conjuntos de tuplas asociados a cada uno de estos ejemplos. La

cardinalidad de cada uno de estos conjuntos varía, mayoritariamente, entre dos y diez, con un máximo de 40 y una media de 4. Cada tupla contiene 166 atributos.

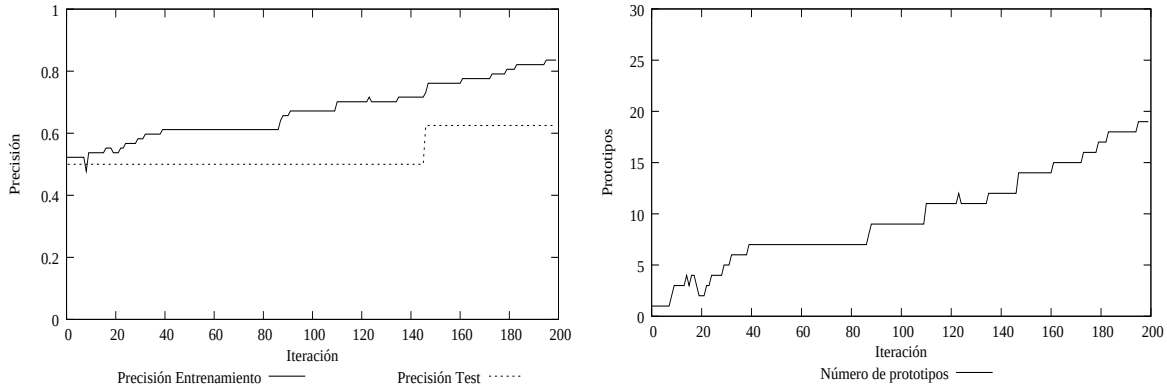


Figura 4.10: Evolución de precisión y número de prototipos con $z = 0{,}0$ en Musk

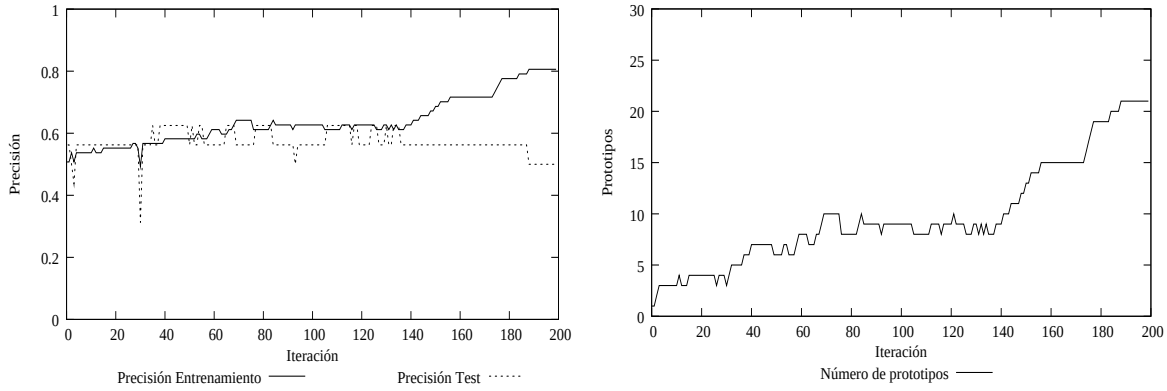


Figura 4.11: Evolución de precisión y número de prototipos con $z = 0{,}5$ en Musk

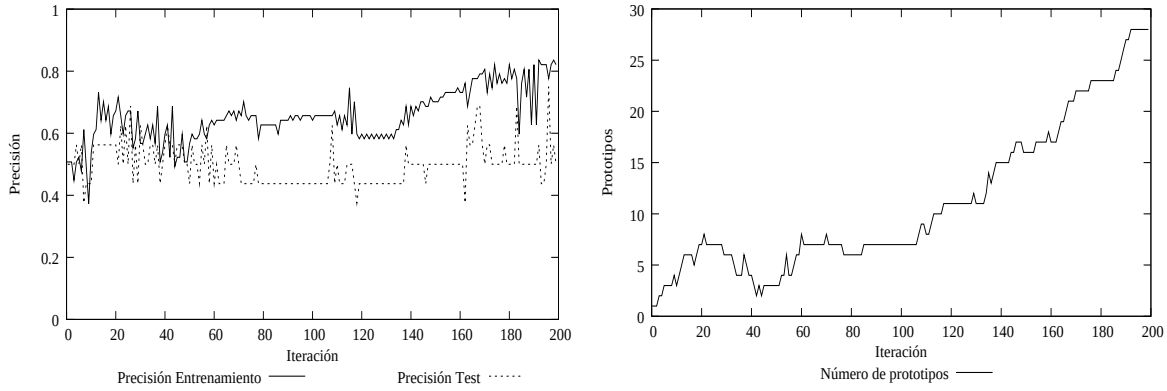


Figura 4.12: Evolución de precisión y número de prototipos con $z = 1{,}0$ en Musk

z	Entr (σ)	Test (σ)	Val (σ)	Prots (σ)	Tiempo (σ)
0,00	54,28 (1,63)	62,71 (0,69)	53,33 (5,19)	3,23 (0,11)	160632,17 (7560,49)
0,10	52,89 (1,66)	60,21 (3,89)	43,70 (5,68)	2,27 (0,31)	155199,70 (5957,00)
0,30	50,71 (1,16)	62,03 (2,58)	51,39 (4,72)	2,65 (0,40)	162897,70 (7628,10)
0,50	55,92 (3,71)	63,75 (2,50)	55,93 (3,95)	4,90 (2,07)	214009,63 (71831,24)
0,70	58,01 (2,62)	63,12 (2,08)	50,37 (7,16)	4,67 (1,36)	215406,87 (76449,42)
0,90	56,42 (1,42)	65,78 (3,59)	51,67 (6,67)	5,35 (1,65)	183157,67 (67268,21)
1,00	62,95 (2,85)	72,81 (1,25)	53,06 (4,31)	10,67 (1,28)	196183,02 (64284,59)

Tabla 4.4: Resultados en Musk

Como puede observarse en la tabla 4.4 y en las figuras 4.10, 4.11 y 4.12, ninguna de las configuraciones del algoritmo obtiene buenos resultados, ya que el porcentaje de éxito en validación se acerca al porcentaje de la clase mayoritaria (cerca del 50 %). A pesar de esto es destacable que observar el cambio en la variación que se produce en las versiones con pesos incorporados, que es capaz de obtener mejores resultados en Test, pero esto seguramente porque el aumento en el número de prototipos, que se debe a un sobreentrenamiento ya que esta mejora no se refleja en los resultados de validación.

4.4.2. Diterpenes

El objetivo de este dominio es identificar el tipo del compuesto de esqueletos compuestos diterpenoides según el resultados de la prueba espectroscópica 13-NMR. Cada ejemplo consta de un conjunto de puntos, cada uno de los cuales se describe mediante su multiplicidad y frecuencia. Es un problema de comparación entre conjuntos. Tiene 1503 instancias y 23 clases. Cada instancias se compone de 20 tuplas.

Como puede observarse en la tabla 4.5, la incorporación de la ponderación de características disminuye la precisión del clasificador obtenido, disminuyendo el número de prototipos. Hay que destacar que esto puede darse porque el número de prototipos no sea lo suficientemente grande, y ya que en en este dominio existe un número grande de clases, seguramente aquellas que no tengan un número suficiente de instancias no estén bien representadas en el clasificador resultante. Cuando $z = 1{,}0$ se llega a la mínima precisión, pero disminuyendo también la varianza, a diferencia de otros dominios donde la varianza para este valor de z aumenta. Seguramente esto pase porque la población no puede crecer de un modo estable y se llegue a la máxima precisión posible con ese número de prototipos.

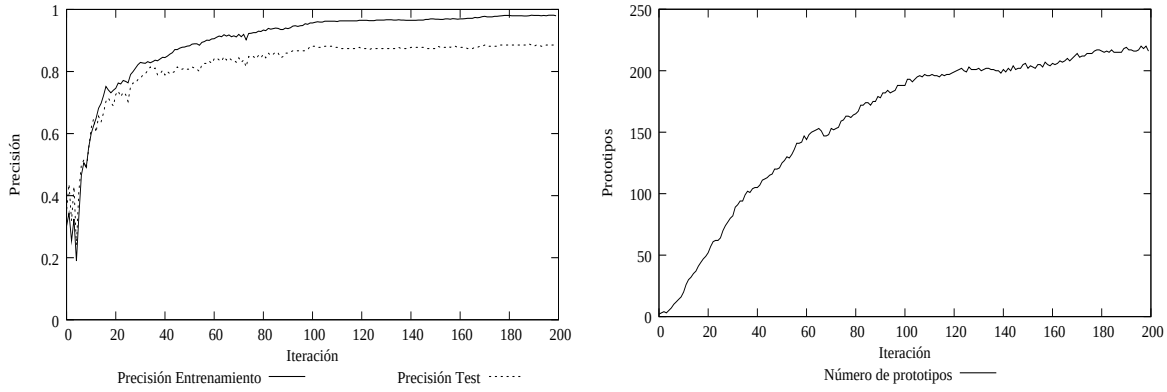


Figura 4.13: Evolución de precisión y número de prototipos con $z = 0{,}0$ en Diterpenes

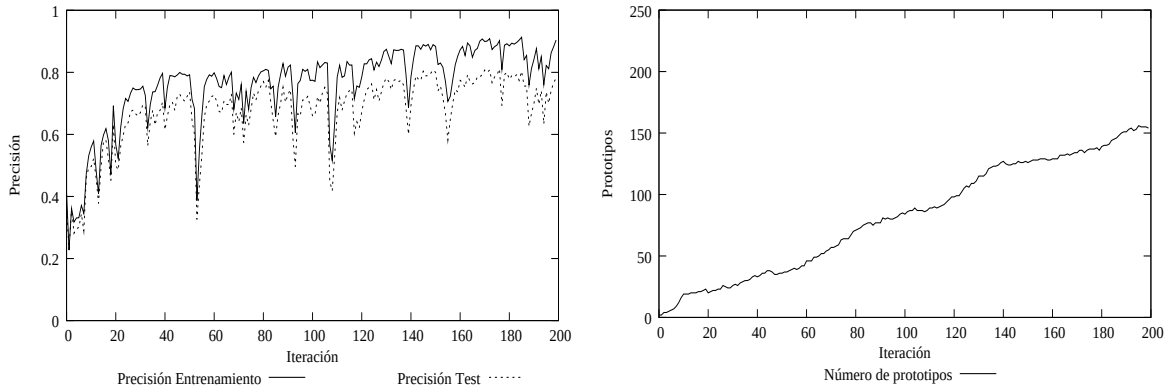


Figura 4.14: Evolución de precisión y número de prototipos con $z = 0{,}5$ en Diterpenes

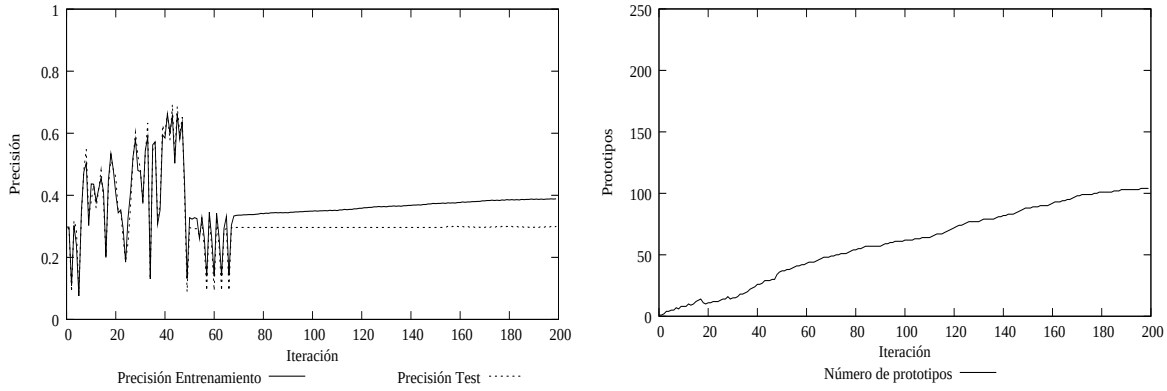


Figura 4.15: Evolución de precisión y número de prototipos con $z = 1{,}0$ en Diterpenes

z	Entr (σ)	Test (σ)	Val (σ)	Prots (σ)	Tiempo (σ)
0,00	96,47 (0,30)	89,47 (0,33)	87,69 (0,65)	195,80 (0,93)	4558256,03 (1189326,51)
0,10	87,84 (1,88)	83,10 (1,59)	80,48 (0,76)	124,68 (22,06)	3268751,00 (1094921,16)
0,30	87,65 (3,49)	83,40 (2,13)	81,53 (1,71)	125,74 (35,47)	3876482,68 (1785251,38)
0,50	85,76 (2,40)	81,62 (1,75)	80,33 (1,65)	113,56 (21,13)	3844251,90 (1697785,28)
0,70	80,93 (0,87)	78,74 (0,74)	76,71 (0,52)	89,07 (8,82)	4182044,87 (1229578,02)
0,90	75,53 (0,55)	74,22 (0,59)	72,40 (1,42)	68,67 (14,82)	3545451,13 (1713133,98)
1,00	60,82 (0,42)	61,29 (0,75)	61,02 (0,78)	23,08 (1,04)	2406357,65 (892091,75)

Tabla 4.5: Resultados en Diterpenes

4.4.3. Mutagenesis

En este dominio se pretende predecir la presencia de mutación en un conjunto de 188 compuestos aromáticos. El dominio se divide en 3 relaciones. En la relación principal cada tupla describe un compuesto distinto (cada uno tiene un identificador único). En la segunda relación se describen los átomos que componenen cada compuesto (entre 1 y 25 átomos por cada compuesto) y en la tercera se describen como están enlazados los átomos entre sí.

z	Entr (σ)	Test (σ)	Val (σ)	Prots (σ)	Tiempo (σ)
0,00	80,15 (1,80)	84,04 (1,54)	78,06 (1,53)	9,15 (1,50)	2192538,10 (305440,25)
0,10	81,15 (2,42)	83,14 (1,83)	78,15 (1,36)	13,00 (2,07)	2998425,60 (251336,27)
0,30	80,59 (1,67)	85,78 (1,44)	76,11 (2,59)	10,13 (1,82)	2613710,10 (654329,87)
0,50	82,08 (1,01)	81,76 (0,98)	77,04 (1,73)	11,67 (1,62)	2939287,70 (277782,53)
0,70	79,46 (1,29)	85,29 (0,98)	76,85 (2,47)	9,60 (1,80)	2336203,10 (493953,40)
0,90	81,27 (0,60)	82,45 (0,65)	75,00 (1,85)	10,93 (0,71)	2844057,27 (265680,56)
1,00	80,34 (1,45)	87,25 (0,85)	77,04 (1,98)	12,63 (3,71)	2644832,57 (315536,36)

Tabla 4.6: Resultados en Mutagenesis

Como se observa en la tabla 4.6los resultados son similares para todos los valores de z. En el segundo caso ($z = 0{,}1$) se observa una ligera mejoría en los resultados de validación, pero en general las diferencias son insignificantes.

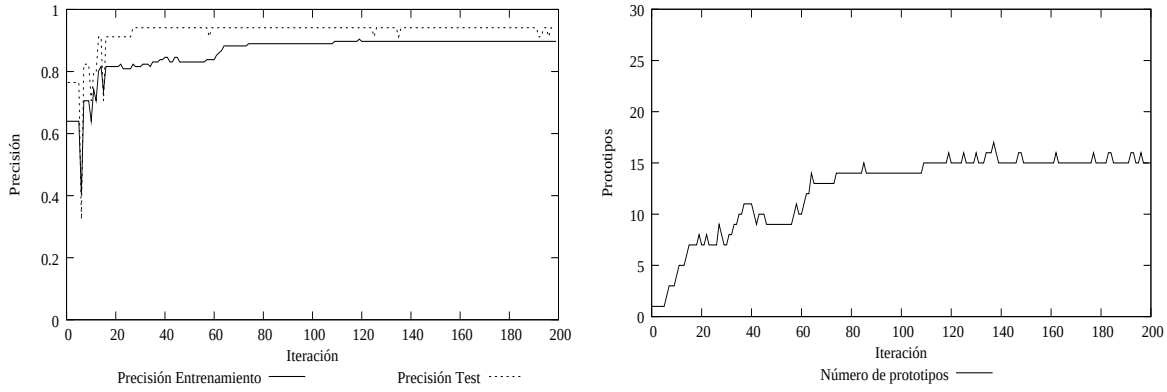


Figura 4.16: Evolución de precisión y número de prototipos con $z = 0{,}0$ en Mutagenesis

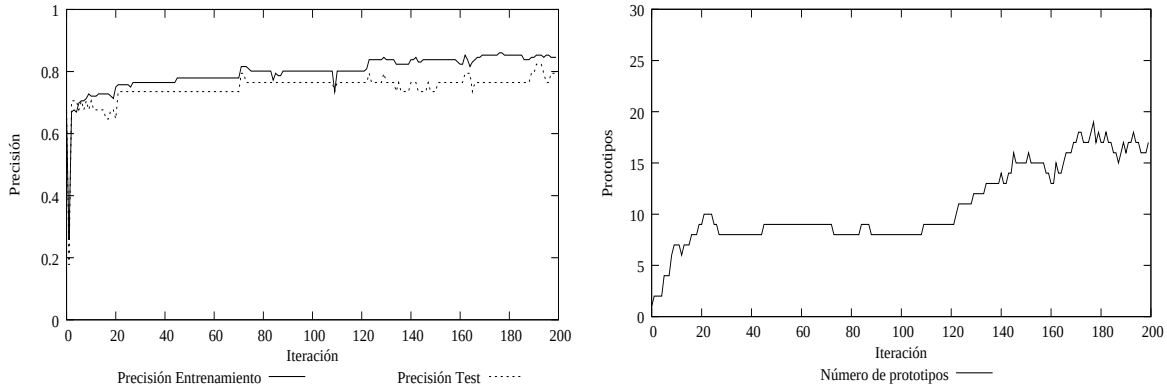


Figura 4.17: Evolución de precisión y número de prototipos con $z = 0{,}5$ en Mutagenesis

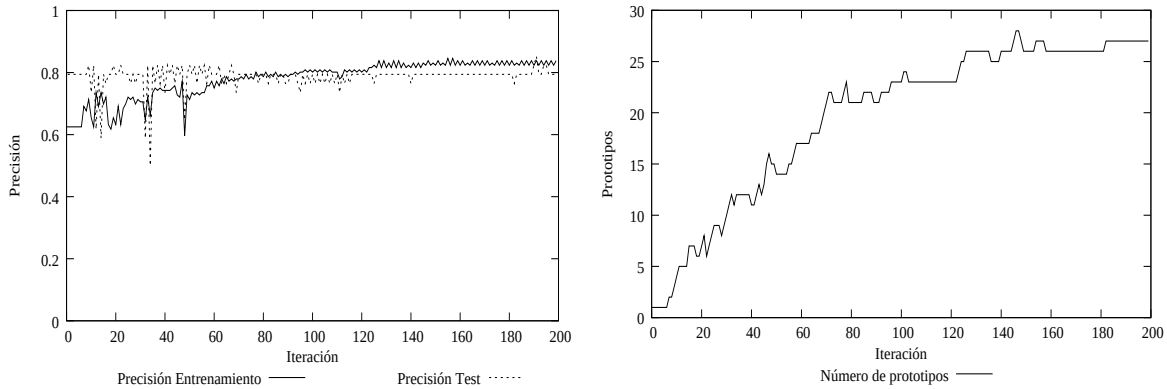


Figura 4.18: Evolución de precisión y número de prototipos con $z = 1{,}0$ en Mutagenesis

4.5. Dominios de planificación

La planificación automática consiste en la resolución de problemas a partir de la especificación de un dominio. El objetivo del aprendizaje en estos casos es descubrir cuál es la acción a aplicar en cada estado para derivar una política que ayude a un planificador a encontrar un plan (clasificar cada estado para saber que acción es la más adecuada). Los datos se han obtenido a partir de problemas de planificación resueltos por el planificador Sayphi [16]. Para cada plan que resuelve un problema se extraen todos los estados por los que se pasa en el plan y las metas a conseguir en el problema, junto con la acción aplicada. Así cada ejemplo se compone de la descripción relacional del estado y las metas del problema, y la clase asociada es la acción que se aplicó.

4.5.1. Blocksworld

Blockworld es un dominio clásico en la investigación en planificación automática. En los problemas de este dominio se pretende llegar a una configuración determinada de la disposición de los bloques. Cada instancia del dominio (un estado del problema) se describe mediante las siguientes relaciones:

- BLOCKSWORLD: la relación principal que describe cada estado y la acción a aplicar.
- ON: describe si un bloque está sobre otro bloque.
- ON TABLE: los bloques que están sobre la mesa.
- ARM EMPTY: si el brazo está libre.
- CLEAR: si un bloque no tiene nigún otro encima.
- ON goal: si en el estado meta un bloque debe estar encima de otro.
- ON TABLE goal : si en estado meta un bloque debe estar encima de la mesa.
- ARM EMPTY goal : si en el estado meta el brazo debe esta libre.
- CLEAR goal: si en el estado meta un bloque no debe estar debajo de otro.

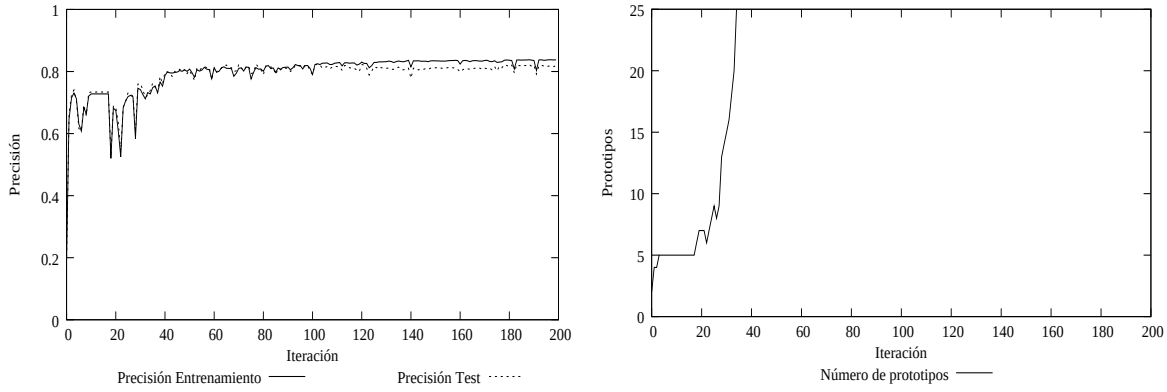


Figura 4.19: Evolución de precisión y número de prototipos con $z = 0{,}0$ en Blocksworld

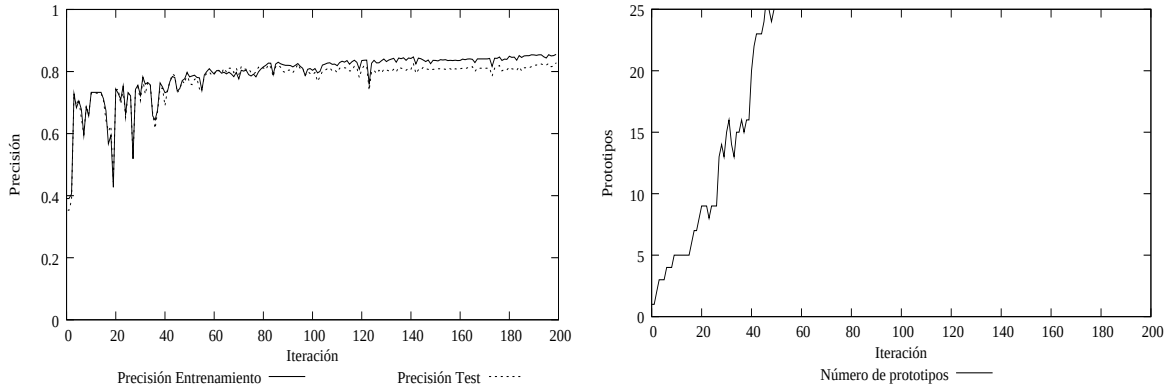


Figura 4.20: Evolución de precisión y número de prototipos con $z = 0{,}5$ en Blocksworld

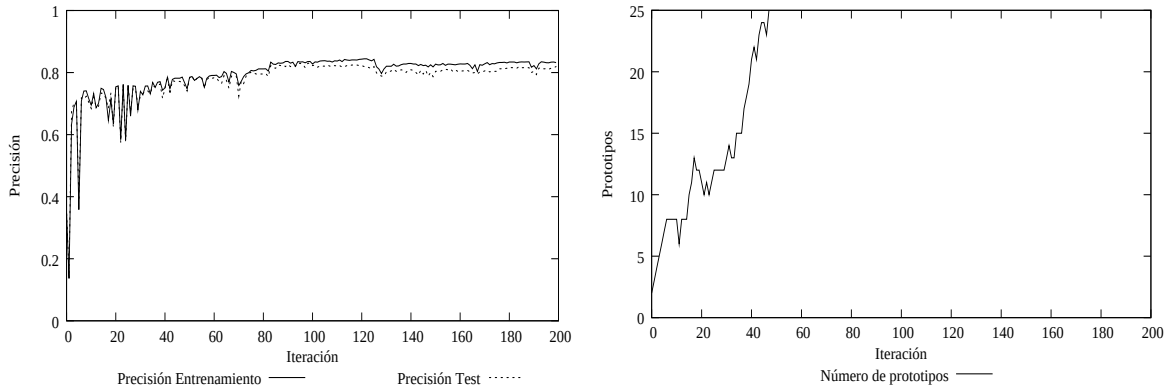


Figura 4.21: Evolución de precisión y número de prototipos con $z = 1{,}0$ en Blocksworld

z	Entr (σ)	Test (σ)	Val (σ)	Prots (σ)	Tiempo (σ)
0,00	83,78 (0,08)	82,30 (0,40)	81,43 (0,15)	107,80 (6,50)	467542,18 (21702,92)
0,10	83,69 (0,27)	82,91 (0,18)	81,83 (0,76)	105,73 (3,07)	450189,70 (22837,95)
0,30	84,10 (0,34)	83,28 (0,67)	81,75 (0,60)	116,58 (6,72)	483731,88 (21569,08)
0,50	83,91 (0,44)	82,53 (0,44)	81,83 (0,33)	106,08 (9,82)	442364,10 (21025,95)
0,70	84,07 (0,66)	82,82 (0,16)	81,92 (0,34)	111,52 (11,71)	455794,08 (31741,32)
0,90	84,12 (0,35)	82,81 (0,05)	81,56 (0,35)	115,22 (4,23)	449238,80 (16105,60)
1,00	83,84 (0,27)	82,19 (0,24)	81,41 (0,35)	109,35 (7,25)	442810,00 (7141,80)

Tabla 4.7: Resultados en Blocksworld

En la tabla 4.7 se observa como la aplicación de pesos no es eficaz en este dominio. Esto se puede explicar porque la ponderación de las tuplas se basa en la diferencia de valores de atributos en las tuplas, pero en estos dominios ejemplos distintos pueden compartir los mismos valores en ciertos atributos, ya que estos valores son identificadores de objetos en el estado de planificación, y que sean iguales en distintos ejemplos no indica que los ejemplos se parezcan más o menos. Por ejemplo si queremos definir un estado en este dominio, donde existe un bloque encima de la mesa podría hacerse utilizando distintos identificadores para el bloque, y seguiría siendo el mismo estado, por ejemplo los dos estados, o instancias relacionales: *ON TABLE bloque1, ON bloque1 bloque2* y *ON TABLE bloque3, ON bloque3 bloque2*; son iguales, aunque los identificadores de los objetos sean distintos.

4.5.2. Zenotravel

En este dominio se busca resolver problemas de logística. Se dispone de vehículos que pueden transportar paquetes entre distintas localizaciones, y, a partir de una serie de localizaciones entre las que los vehículos pueden transitar, el objetivo es que una serie de que cada mercancía se encuentre en una de estas localizaciones. La descripción de cada caso se divide en las siguientes relaciones:

- ZENOTRAVEL: es la relación principal, donde cada instancias se describe mediante un iterador y la acción asociada.
- AT : describe donde esta un objeto en el estado actual.
- FUEL-LEVEL: describe el nivel de combustible de un vehículo.
- IN: describe dentro de dónde se encuentra un paquete que hay que entregar.

- AT goal: describe dónde debe estar un objeto en el estado final (en que lugar debe entregarse).

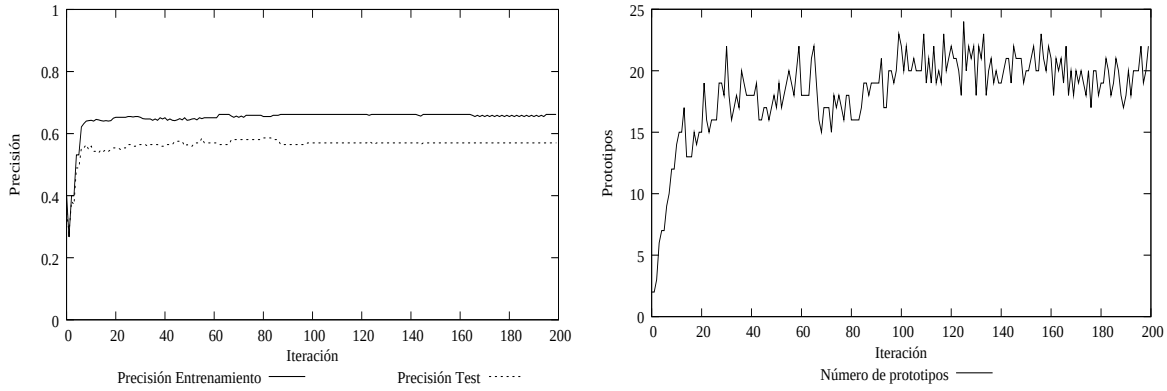


Figura 4.22: Evolución de precisión y número de prototipos con $z = 0{,}0$ en Zenotravel

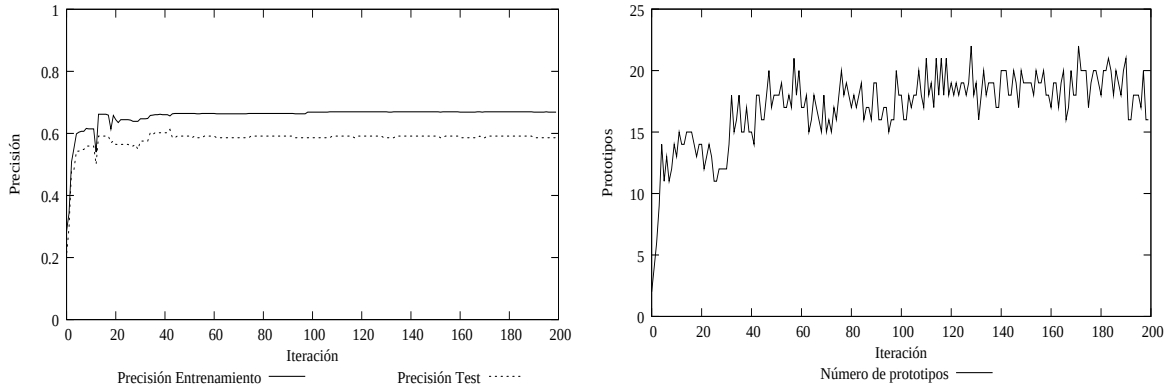


Figura 4.23: Evolución de precisión y número de prototipos con $z = 0{,}5$ en Zenotravel

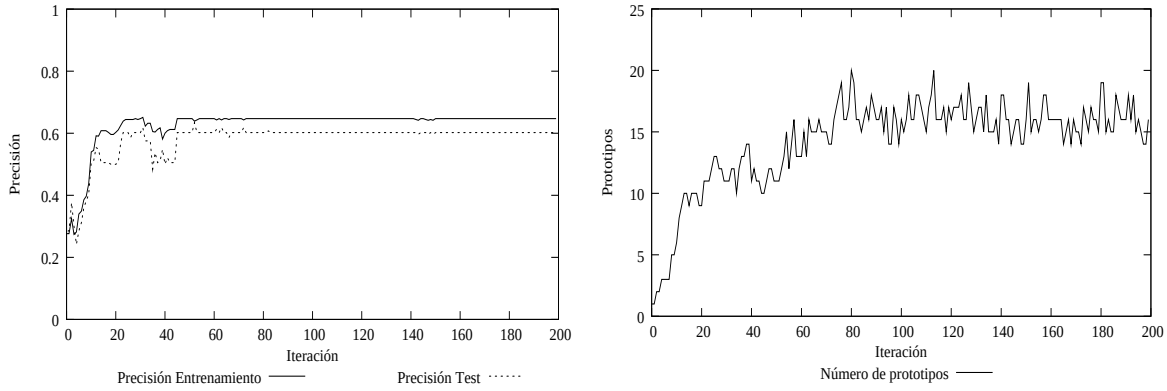


Figura 4.24: Evolución de precisión y número de prototipos con $z = 1{,}0$ en Zenotravel

z	Entr (σ)	Test (σ)	Val (σ)	Prots (σ)	Tiempo (σ)
0,00	63,81 (0,52)	60,77 (0,25)	58,66 (0,68)	14,82 (0,66)	31389,20 (5948,48)
0,10	63,73 (0,61)	61,28 (1,11)	58,03 (0,75)	16,20 (1,10)	35618,38 (1481,34)
0,30	63,77 (0,29)	60,86 (0,89)	58,33 (0,46)	14,48 (0,12)	36061,22 (1514,98)
0,50	63,93 (0,33)	61,39 (0,87)	58,76 (0,68)	15,56 (0,49)	32546,94 (6764,07)
0,70	64,00 (0,63)	60,91 (0,51)	59,13 (0,58)	14,80 (1,30)	35927,80 (2547,40)
0,90	63,17 (0,39)	60,30 (0,40)	58,01 (0,39)	15,05 (0,70)	35408,12 (2211,21)
1,00	63,73 (0,55)	60,49 (0,80)	58,72 (0,72)	14,18 (0,74)	31774,14 (5698,43)

Tabla 4.8: Resultados en Zenotravel

Al igual que en el dominio Blocksworld en este caso no existen diferencias para distintas configuraciones de pesos. Además de los inconvenientes comentados en ese caso en este caso es muy difícil poder aprender un clasificador de los datos de entrada, ya que la mayoria de las instancias comparten tuplas idénticas en varias relaciones.

4.6. Resumen de resultados

En esta sección se resumen de los resultados obtenidos a lo largo de las pruebas y conclusiones sobre éstos.

A la luz de los resultados obtenidos se pueden establecer las siguientes conclusiones:

- En algunos casos utilizar únicamente la distorsión de las regiones asociadas a la iteración actual ($z = 1{,}0$) puede hacer que se consigan clasificadores con gran calidad en pocas iteraciones, aunque afecte notablemente a la convergencia de la población.
- La actualización de pesos tiende en algunos casos a reducir el número de prototipos. Esto puede suponer que el algoritmo generalice mejor en algunos casos (como en el dominio 3), y puede ser perjudicial en otros, como en el dominio Diterpenes.
- El tiempo consumido por el algoritmo es muy elevado. La necesidad de recalcular todas las distancias y el valor de la distorsión en cada nueva iteración y el alto coste del cálculo de la distancia son las principales razones. En este caso la reducción de prototipos es ventajosa, pero nunca lo suficiente como para que compense en tiempo de cómputo.

- La diferencia entre los dominios artificiales y los dominios reales es grande. Aunque estructuralmente son parecidos (ninguno tiene muchas relaciones, y generalmente cada clave ajena se corresponde con una única relación). Obviamente los dominios artificiales estan diseñados para probar la eficacia de la ponderación de la distancia. Seguramente los dominios reales no mantienen una estructura de regiones tan clara como la creada para los dominios artificiales.

- En los dominios de planificación la ineficacia del algoritmo se puede explicar por otras razones. La principal es que la ponderación de las tuplas se basa en la diferencia de valores de atributos en las tuplas que componen cada instancia. Sin embargo, los valores de los atributos en estas relaciones no indican una similitud en las tuplas, ya que son indentificadores de objetos, elegidos arbitrariamente.

- La ponderación de pesos es capaz de asignar pesos a los atributos para discrimnar en las distintas regiones, llegando en algunas ejecuciones a la configuración óptima de prototipos.

- En el inicio del proyecto se estableció la hipótesis de que la incorporación de pesos podría ser efectiva sobre todo en dominios donde la diferencia entre el porcentaje de éxito en entrenamiento y validación fuese grande en RNPC. Esta diferencia se consiguió reducir para los dominios 2 y 3.

- En la tabla 4.9 se muestra un resumen de los resultaos para todos los dominios. En negrita se resaltan los mayores valores para las medidas de éxito de clasificación en entrenamiento (En), éxito de clasificación en test(Te) y éxito de clasificación en validación(Val), además de los menores números de prototipos conseguidos para cada clasificador (Pr).

		Dom 1	Dom 2	Dom 3	Mutagenesis	Diterpenes	Musk	Blocksworld	Zenotravel
N° Instancias		200	200	200	188	1503	92	3260	1034
N° Clases		2	2	2	2	23	2	4	3
	En(σ)	75,56(3,72)	91,50(0,45)	96,66(0,53)	80,15(1,80)	**96,47**(0,30)	54,28(1,63)	83,78(0,08)	63,81(0,52)
	Te(σ)	68,75(0,97)	90,43(0,49)	95,54(0,56)	84,04(1,54)	**89,47**(0,33)	62,71(0,69)	82,30(0,40)	60,77(0,25)
$z = 0$	Val(σ)	53,63(1,13)	89,83(0,56)	93,06(1,19)	78,06(1,53)	**87,69**(0,65)	53,33(5,19)	81,43(0,15)	58,66(0,68)
	Pr(σ)	41,18(5,74)	10,40(0,53)	33,12(2,54)	**9,15**(1,50)	195,80(0,93)	3,23(0,11)	**107,80**(6,50)	14,82(0,66)
	En(σ)	69,65(2,29)	93,47(0,83)	99,14(0,13)	81,15(2,42)	87,84(1,88)	52,89(1,66)	83,69(0,27)	63,73(0,61)
	Te(σ)	65,14(0,14)	96,02(1,36)	99,61(0,15)	83,14(1,83)	83,10(1,59)	60,21(3,89)	82,91(0,18)	**61,28**(1,11)
$z = 0{,}1$	Val(σ)	51,50(3,50)	92,50(0,33)	98,54(0,61)	**78,15**(1,36)	80,48(0,76)	43,70(5,68)	81,83(0,76)	58,03(0,75)
	Pr(σ)	27,45(1,85)	7,50(0,87)	10,20(0,33)	13,00(2,07)	124,68(22,06)	**2,27**(0,31)	105,73(3,07)	16,20(1,10)
	En(σ)	72,50(2,15)	94,10(0,60)	99,13(0,19)	80,59(1,67)	87,65(3,49)	50,71(1,16)	**84,10**(0,34)	63,77(0,29)
	Te(σ)	66,94(0,28)	**97,04**(0,80)	99,19(0,34)	85,78(1,44)	83,40(2,13)	62,03(2,58)	**83,28**(0,67)	60,86(0,89)
$z = 0{,}3$	Val(σ)	53,00(0,50)	92,67(1,11)	97,92(0,78)	76,11(2,59)	81,53(1,71)	51,39(4,72)	81,75(0,60)	58,33(0,46)
	Pr(σ)	29,10(1,50)	7,10(0,60)	**8,60**(0,27)	10,13(1,82)	125,74(35,47)	2,65(0,40)	116,58(6,72)	14,48(0,12)
	En(σ)	78,61(2,15)	**94,77**(0,49)	99,13(0,03)	82,08(1,01)	85,76(2,40)	55,92(3,71)	83,91(0,44)	63,93(0,33)
	Te(σ)	66,39(0,00)	96,20(1,05)	99,21(0,25)	81,76(0,98)	81,62(1,75)	63,75(2,50)	82,53(0,44)	61,39(0,87)
$z = 0{,}5$	Val(σ)	57,75(2,25)	92,33(1,22)	97,58(1,22)	77,04(1,73)	80,33(1,65)	**55,93**(3,95)	81,83(0,33)	58,76(0,68)
	Pr(σ)	39,45(2,15)	9,53(0,98)	9,03(0,18)	11,67(1,62)	113,56(21,13)	4,90(2,07)	106,08(9,82)	15,56(0,49)
	En(σ)	72,15(1,87)	93,52(0,90)	99,20(0,14)	79,46(1,29)	80,93(0,87)	58,01(2,62)	84,07(0,66)	**64,00**(0,63)
	Te(σ)	65,00(1,67)	96,39(0,56)	99,19(0,12)	85,29(0,98)	78,74(0,74)	63,12(2,08)	82,82(0,16)	60,91(0,51)
$z = 0{,}7$	Val(σ)	56,00(1,00)	**94,50**(0,33)	98,29(1,36)	76,85(2,47)	76,71(0,52)	50,37(7,16)	**81,92**(0,34)	**59,13**(0,58)
	Pr(σ)	28,60(3,10)	8,33(0,89)	8,97(0,49)	9,60(1,80)	89,07(8,82)	4,67(1,36)	111,52(11,71)	14,80(1,30)
	En(σ)	77,60(0,31)	93,26(0,46)	**99,54**(0,11)	81,27(0,60)	75,53(0,55)	56,42(1,42)	84,12(0,35)	63,17(0,39)
	Te(σ)	68,33(0,28)	96,20(0,99)	**99,83**(0,03)	82,45(0,65)	74,22(0,59)	65,78(3,59)	82,81(0,05)	60,30(0,40)
$z = 0{,}9$	Val(σ)	56,50(1,00)	92,67(1,11)	**99,44**(0,19)	75,00(1,85)	72,40(1,42)	51,67(6,67)	81,56(0,35)	58,01(0,39)
	Pr(σ)	39,00(0,50)	**6,93**(0,98)	9,15(0,15)	10,93(0,71)	68,67(14,82)	5,35(1,65)	115,22(4,23)	15,05(0,70)
	En(σ)	**83,96**(2,92)	91,30(1,88)	93,15(6,87)	80,34(1,45)	60,82(0,42)	**62,95**(2,85)	83,84(0,27)	63,73(0,55)
	Te(σ)	**83,89**(0,28)	94,10(1,42)	93,40(6,06)	**87,25**(0,85)	61,29(0,75)	**72,81**(1,25)	82,19(0,24)	60,49(0,80)
$z = 1{,}0$	Val(σ)	**75,00**(4,00)	88,25(2,50)	92,53(6,72)	77,04(1,98)	61,02(0,78)	53,06(4,31)	81,41(0,35)	58,72(0,72)
	Pr(σ)	**22,10**(3,40)	7,10(1,90)	14,42(2,98)	12,63(3,71)	**23,08**(1,04)	10,67(1,28)	109,35(7,25)	**14,18**(0,74)

Tabla 4.9: Resumen de resultados

Capítulo 5

Conclusiones y trabajo futuro

5.1. Conclusiones sobre los objetivos iniciales del proyecto

En este punto se enumeran y explican cuáles fueron los principales requisitos y esfuerzo dedicado en el desarrollo de los objetivos impuestos en el proyecto.

En el anexo B se puede observar el trabajo y coste invertido en cada uno de estos objetivos.

1. Diseño e implementación del algoritmo LFW-RNPC

   a) *Estado de la cuestión*
   La literatura examinada para este trabajo se consideró suficiente, y las fuentes de información para la resolución de dudas fueron fáciles de encontrar. La dificultad del material no era grande aunque en varias ocasiones se debió consultar a expertos en el campo, para asegurar la compresión de varios conceptos.

   b) *Estudio de las APIs de Weka y Relational Weka, y de la implementación del algoritmo RNPC*
   En esta fase se detectaron varias deficiencias en la plataforma Relational Weka, que determinaron la ampliación del trabajo de desarrollo que se realizó. Además tuvo que replanterase el alcance del proyecto, debido a que la carga en la parte de desarrollo debía ser mayor a la esperada en un principio.

c) *Diseño de la medida de distancia relacional*
Tras la lectura de la literatura, este objetivo implicó la toma de varias decisiones de diseño en la medida de distancia propuesta. La dificultad es que la justificación de estas decisiones no se pudo realizar con dominios reales, sino con casos de prueba.

d) *Diseño y elección del método de actualización de pesos*
La adaptación de la medida de la distorsión implicó una decisión importante para los resultados (la omisión de pesos en claves ajenas). Para incorporar la actualización de pesos fue muy importante el repaso de la literatura.

e) *Diseño de componentes*
Este objetivo no tuvo complicaciones especiales, excepto alguna modificación en el diseño realizada en el comienzo de la implementación.

f) *Implementación del algoritmo*
A pesar de la complicación de algunos pasos del algoritmo y de la complicación técnica de algunos componentes, este objetivo no supuso una dificultad excesiva respecto a otros. Esta fase estuvo continuamente ligada con la fase de validación y verificación.

g) *Validación y verificación del algoritmo*
Este objetivo fue bastante costoso, debido a que varios errores del algoritmo se detectaron en la ejecución de los dominios más grandes, lo que incrementó el tiempo hasta asegurar la corrección del sistema implementado.

2. Experimentación y evaluación de LFW-RNPC

a) *Definición y selección de los dominios*
Como se explicó en el punto 4, se requirió la creación a de varios dominios artificiales, para la comprobación de la validez del algoritmo. No pudieron encontrarse muchos conjuntos de datos reales relacionales que pudiesen utilizarse, debido a que en algunos era infactible la experimentación, por el tiempo, y otros tenían una estructura relacional demasiado compleja para ser procesados por los componentes especificados en el diseño.

b) *Experimentación*
Cabe destacar que esta fue la fase más larga del proyecto. El tiempo invertido se debió principalmente al tiempo de ejecución de LFW-RNPC.

Las principales dificultades que surgieron para completar este objetivo, fueron los errores del algoritmo que se detectaron en esta fase.

c) *Análisis de los resultados*
El examen de los resultados, comprendió una etapa muy larga del proyecto. Esto es así porque esta fase se acopló en cierta medida con la validación del algoritmo, al constrastar los resultados esperados en los dominios artificiales con los resultados obtenidos.

3. Redacción de la memoria del proyecto
La principal dificultad se encontró en el resúmen de la literatura previa, y en la muestra y explicación de los resultados.

5.2. Conclusiones sobre la ponderación de pesos en dominios relacionales

Tras el análisis a posteriori de los resultados experimentales se llegó a dos conclusiones principales. La primera conclusión tiene que ver con la actualización de los pesos asociados a las claves ajenas. La decisión de mantener estos pesos constantes (como se explica en la sección 3.3) cobró relevancia en los dominios de planificación (los resultados obtenidos en estos dominios se muestran en la sección 4.5). En estos dominios podría ser importante saber si una instancia tiene tuplas asociadas en una relación o no. En la medida de distancia propuesta sólo se tiene en cuenta si existen o no.

La segunda conclusión es que el uso del medoide como representante de la región con la inclusión de la ponderación de pesos dificulta la convergencia. La hipótesis considerada para explicar este hecho es que el medoide de la región puede desplazarse mucho entre iteraciones, lo que dificulta que la configuración de pesos asociados a una región se estabilice.

5.3. Conclusiones sobre los resultados experimentales

Según los resultados explicados el el capítulo 4, la adición de la ponderación local tal y como se ha planteado, no aporta un mejora relevante a los resultados obtenidos en dominios experimentales.

Ya que el efecto sí es notable en los dominios artificiales creados a propósito para estudiar el efecto obtenido en la ponderación, debe considerarse qué diferencias existen entre los dominios artificiales y los experimentales.

Los dominios artificiales se crearon para estimar la capacidad del algoritmo para asignar prototipos a las diferentes regiones, definiendo cada región a partir de unos atributos relevantes con unos valores determinados. En estos casos se demostró que LFW-RNPC podía mejorar la aproximación que no tenía en cuenta la ponderación.

Sobre los resultados experimentales obtenidos en los dominios experimentales pueden añadirse las siguientes conclusiones:

- Las aproximaciones basadas en instancias no funcionan en el dominio, por lo tanto la incorporación de la ponderación local tampoco puede aportar una mejora sustancial (esto es lo que se observa en los resultados obtenidos en el dominio MUSK, que se exponen en la sección 4.4.1).
- La incorporación de la ponderación de pesos tiende a reducir el número de prototipos en algunos dominios. Por lo tanto si un dominio tiene muchas clases o una distribución de ejemplos que haga difícil que la selección de instancias de entrada sea lo suficientemente representativa como para que grupos pequeños de clases minoritarias puedan tener un prototipo propio. Esto se deduce de los resultados obtenidos en el dominio Diterpenes (sección 4.4.2).
- La decisión de mantener constante el peso asociado a claves ajenas hace que el desarrollo del algoritmo con la ponderación local de características sea igual al caso sin ponderación de pesos. Este es el caso de los dominios de planificación, los cuales son dominios con varias relaciones, pero donde el número de atributos que no son claves ajenas en cada relación no es muy grande. Además en estos dominios, ya que los valores de cada tupla son identificadores que designan objetos del problema y no valores descriptivos de la instancia, el peso asociado a cada atributo en realidad es irrelevante.

5.4. Trabajo futuro

5.4.1. Distancia y cálculo de la distorsión

La principal opción en este apartado es estudiar variaciones en la medida de distancia. Debido al tiempo que hubiera requerido comparar varias medidas entre sí, en este proyecto se optó por concentrarse en una única medida de distancia.

Entre las variaciones posibles en la medida de distancia propuesta se encuentran:

- Cambio de la medida de distancia entre conjuntos: se optó por la distancia entre conjuntos de RIBL, pero podrían realizarse experimentos con otras distancias entre conjuntos de entre las explicadas en el punto 2.2.3.
- Cambio de distancia entre tuplas: en este punto podría considarse el efecto de la normalización sobre atributos numéricos.
- Otro elemento importante en la definición de la función de distancia es la definición de la distorsión asociada a las caractrísticas que son referenciadas por claves ajenas.

En cuanto al cálculo de la distorsión también sería posible considerar características distintas a la distancia media en la región, por ejemplo podría considerarse la desviación estándar de la distancia media al representante de la región, ya que seguramente un atributo con una desviación muy grande sea poco relevante en una región, lo que podría reflejarse de alguna manera en su peso asociado.

5.4.2. Representante de la región

Como se explicó en el capítulo 3, el representante de la región en este algoritmo es el medoide de la región. Según las conclusiones sobre los experimentos, el uso del medoide puede ser una de las razones que afecten negativamente a los resultados del algoritmo.

Un posible vía futura consistiría en el cambio del representante de la región de un medoide por un centroide. Esto implicaría proponer un nuevo método para la selección del centroide relacional.

5.4.3. Optimización del algoritmo

Como se ha señalado anteriormente uno de los principales inconvenientes que ha habido en el desarrollo del proyecto ha sido el tiempo de ejecución que requiere LFW-RNPC. A pesar de que este tiempo se ha intentado mejorar en la medida de lo posible (paralelizando varias fases del algoritmo como se muestra en el anexo A).

Entre las principales opciones para esto se proponen el cacheo de distancias entre iteraciones consecutivas (obtener un política de cacheo para no tener que recalcular las distancias de una región en caso de que no haya habido un transpaso de instancias entre esa región y otra, por ejemplo).

Como se ha comentado en el punto anterior, quizá utilizar un centroide como representante pudiera ser menos costoso, dependiedo del algoritmo que se utilice para ello.

5.4.4. Flujo de ejecución

En este apartado se podrían estudiar criterios de finalización distintos al número de iteraciones. Por ejemplo podría estudiarse alguna medida de variación de las regiones entre iteraciones, o a la variación de los pesos asociados a la población.

5.5. Conclusiones generales

El principal inconveniente en el desarrollo del proyecto ha sido que la necesidad de centrarse en aspectos técnicos ha limitado la dedicación al desarrollo de la parte de la investigación. Una de las principales razones por las que no han podido obtenerse unos resultados mejores en la experimentación es que el trabajo y tiempo requerido para ello en términos de experimentación sería inabarcable dentro de este proyecto.

A pesar de los inconvenientes se ha conseguido desarrollar un sistema de ponderación local de pesos para dominios relacionales, no conociéndose aproximaciones similares relevantes en el estado del arte. Además el software desarrollado y las ideas propuestas en este proyecto, asientan una base sobre la que podría basarse un trabajo futuro, que pudiese conseguir resultados más satisfactorios que los obtenidos en el presente trabajo.

Anexo A

Diseño y desarrollo

En este anexo se comentan aspectos técnicos del desarrollo del proyecto. Debido a que se considera que la parte central de este proyecto se encuentra en la parte de investigación, no se expondrán detalladamente los requisitos y soluciones técnicas propuestas.

A.1. Requisitos de la implementación

A continuación se enumeran informalmente los requisitos que debe cumplir la implementación del sistema que permita la experimentación con el algoritmo LFW-RNPC.

A.1.1. Selección de tecnología para el desarrollo

Debido a que el proyecto parte de los servicios ofrecidos por las plataformas Weka y Relational Weka, que están implementadas en JAVA, se ha elegido este lenguaje para facilitar el uso de las APIs ofertadas por estas plataformas. Además se ha utilizado el entorno de desarrollo Eclipse, en su versión Europa.

La memoria se redactó con el sistema Latex (pdfTeX 3.1415926-1.40.10-2.2), las gráficas de resultados se generaron con el programa GNUPlot (versión 4.2), los diagramas UML, que aparecen en la segunda sección de este anexo, se generaron con el programa ArgoUMLv0.30.2, y el diagrama de Gantt del anexo B se generó con el programa OpenProject.

La generación de los dominios artificiales y parseo de los dominios de planificación se realizó mediante scripts en lenguaje Ruby.

A.1.2. Generación y obtención de dominios

En el caso de los dominios artificiales que se explican en el capítulo 4, la generación se realizó definiendo los esquemas manualmente y luego generando un número de instancias de acuerdo a las regiones que querían definirse.

Los dominios Musk, Diterpenes y Mutagenesis fueron descargados de la página de Relational Weka [1].

Los dominios de planificación fueron transformados a archivos ARFF a través de ejemplos resultantes de la solución de esos problemas con el planificador Sayphi [16]. La transformación de los archivos se realizó mediante scripts en Ruby.

A.1.3. Gestión de ejemplos

Como se describió en el capítulo 3, LFW-RNPC necesita recalcular las distancias en función de los pesos asociados a cada prototipo en cada iteración. Esto requiere que se acceda a los ejemplos en la fase del cálculo de distancias y la actualización de pesos. Este hecho implica que no se pueda reutilizar la gestión de ejemplos que se realiza en Relational Weka por los siguientes motivos:

- Relational Weka implementa algoritmos relacionales basados en instancias, que utilizan medidas de distancias que no varían durante la ejecución del algoritmo. Para optimizar el tiempo de ejecución, cuando se ejecuta un algoritmo basado en instancias relacional primero se precalcula un matriz de distancias que se almacena en memoria y es consultada durante la ejecución.
- Sólo se consultan los valores de las tuplas de las distintas relaciones cuando se calcula la matriz de distancias. La búsqueda de tuplas en función de la clave de una relación es lineal (recorre todas las tuplas de una relación comparando el valor de cada tupla el valor buscado).
- Si se utilizase este modelo de gestión de ejemplos el tiempo de cada ejecución de LFW-RNPC sería muy grande, ya que es imposible almacenar las distancias

[1] http://cui.unige.ch/ woznica/rel_weka

debido a que los pesos asociados a cada región varían, y a que la región a la que pertence cada instancia varía entre iteraciones. Por esto es necesario recalcular la matriz de distancia entre iteraciones.

En LFW-RNPC se realiza una fase de preprocesado del esquema relacional donde se indizan las instancias de cada relación por los valores de sus claves ajenas, de tal manera que el acceso a las tuplas en cada relación que corresponden a una instancia relacional es constante. Así, para cada valor de una clave ajena se conocen las tuplas de la relación que tienen ese valor en el atributo que es clave ajena. Este paso mejora sustancialmente el tiempo de cálculo de la distancia entre atributos que son claves ajenas (la que se describe en la fórmula (3.1) del capítulo 3).

A.1.4. RNPC

En el inicio del proyecto se esperó poder reutilizar el flujo de ejecución del algoritmo en una implementación de RNPC. El problema principal es que esta implementación precalculaba la matriz de distancias, y no incorporaba los componentes necesarios para la gestión de los prototipos.

Tras examinar el código existente se llegó a la conclusión de que su reutilización sería muy costosa y se decidió reimplementar los componentes de RNPC necesarios para LFW-RNPC.

A.1.5. Optimización del tiempo de ejecución

Como se ha señalado anteriormente la ejecución de LFW-RNPC consume una gran cantidad de tiempo respecto a otros algoritmos. Por lo tanto se han intentado optimizar la ejecución del algoritmo mediante la paralelización de dos fases: el cálculo del medioide que se realiza en la fase de selección del representante de cada región, y el cálculo del prototipo más cercano a cada instancia que se hace en la fase de asignación de instancias. Ya que en la máquina en la que se realizaron los experimentos se disponía de dos hilos de ejecución simultáneos, para cada fase se definió un esquema de reparto de carga, para equilibrar el trabajo realizado por cada hilo:

- En el cálculo del medoide se requería calcular una matriz simétrica y de diagonal nula, donde se recogiesen todas las distancias entre pares de instancias de la

región. En este caso el esquema de reparto de carga es bastante sencillo. Como la matriz es simétrica, en cada fila se cálcula una posición menos que en la anterior (en la primera se calculan $n-1$ posiciones en la siguiente $n-2$, hasta la penúltima, donde sólo se calcula una).

- En la búsqueda del prototipo más cercano se tiene que calcular la distancia entre cada instancia y cada prototipo. Para realizar este cálculo se dividen las instancias en dos conjuntos disjuntos, y para cada uno de ellos un hilo distinto calcula el protipo más cercano a cada instancia.

A.1.6. Evaluación de resultados

Se ha implementado un sistema de validación propio, ya que utilizar los sistemas de validación implementados en RelationalWeka hubiera requerido cambios importantes en la implementación de los sistemas de gestión de ejemplos y de consulta de distancias respecto a un clasificador.

Por lo tanto se implementó una clase que llevase a cabo las siguientes tareas:

- División de ejemplos: debe realizarse la distribución de todas los ejemplos de entrada en un número de cubetas igual a un número determinado por parámetro. Cada cubeta está compuesta por ejemplos seleccionados al azar, hasta llegar a un número determinado.
- Validación: Cuando se obtiene el clasificador de cada iteración de la validación cruzada debe validarse respecto a la cubeta correspondiente.

A.1.7. Registro de resultados

Para el análisis de los resultados y para la validación y verificación del sistema debe conocerse cierta información de la ejecución, que es la siguiente:

- Traza de la ejecución: esta información comprende una serie de características que definen cada iteración. Estas características son:
  - Tiempo de ejecución (en milisegundos).
  - Tamaño de la población al final de la iteración.

  - Precisión obtenida en el conjunto de entrenamiento.
  - Precisión obtenida en el conjunto de test.
  - Número de prototipos que han muerto.
  - Número de reproducciones.
  - Número de luchas.

- Clasificador: necesita registrarse el clasificador resultante de la ejecución. Se decidió utilizar dos tipos de registro:
  - Registro textual: donde cada prototipo de la población se representa como texto, y donde para cada uno se puede observar su medoide, el número de instancias de entrenamiento de cada clase contenidas en su región y el peso asociado a cada atributo de cada relación.
  - Registro binario: donde se guardan los objetos correspondientes a los prototipos de la población para cargarlos más fácilmente desde un programa JAVA.
- Resultados de la validación: para cada validación se almacena un resumen de los resultados obtenidos en cada cubeta y los parámetros con los que se inició la ejecución del algoritmo.

A.1.8. Gestión de parámetros del algoritmo

Debido al número de parámetros del sistema (los del algoritmo propiamente, y otros referidos a gestión de rutas, registros de resultados, etc) era relativamente grande, se decidió que la entrada de configuración se encontrase en un archivo de texto. Este archivo se encuentra formateado en pares *clave* : *valor*, fácilmente legible desde un programa en JAVA.

A continuación se incluye un ejemplo:

```
#cargar Algoritmo, 1 = SI, 0 o nada = NULL
cargar = 0
#nombre del archivo del algoritmo que va a cargarse
archivoAlgoritmo =
#numero de iteraciones entre cada guardado del algoritmo
#si es menor a 1 no se guarda
```

```
iteracionesGuardado =  -1
#maximo de prototipos (-1 -> sin limite) ,
maxPrototipos = -1
#iteraciones del algoritmo [1,..]
iteraciones = 200
#profundidad maxima de la distancia [1,..]
profundidad = 5
#numero de folds para la validacion [1,..]
folds = 10
#Porcentaje de instancias del total de
#instancias de entrenamiento para la validacion en el algoritmo
validacionAlgoritmo = 20
#factor de mdificacion para la actualizacion de los pesos [0,1]
fMod = 1.0
#fichero mtarff del que se carga el dominio
mtarff = raiz.mtarff
#fichero .matriz del que se cargan las distancias
#(en el caso de que se precalculen)
matriz = matrices/mutagenesis_v1.matrix
#indica si se quiere actualizar lso pesos o no (si/no)
actualizarPesos = si
#indica si se quiere cargar los pesos (si/no)
cargarPrototipos = no
#indica el archivo del que se cargan los prototipos,
#en caso de que estuviese activa la opcion
archivoPrototipo = prototipos
#Indica si se quiere guardar la traza o no
traza = si
#Archivo de resultados. Imprime los resultados de cada cubeta.
archivoResultados = resultados.txt
#Directorio donde se guardan los clasificadores y los resultados
dirGuardado = resultados3
#Si se activa este parametro la distancia entre instancias se
#normaliza entre la suma de los pesos
normalizarSumaPesos = si
#Si se activa este parametro la distancia entre instancias se
```

```
#normaliza entre el numero de atributos de la relacion
normalizarAtributos = no
#Si se activa este parametro la distorsion de un prototipo solo se calcula
#con las instancias en su region que sean de la misma clase
distorsionRegionClase = no
#nombre del archivo de log, si el campo es vacio no se guarda ningun log
archivoLog = log.txt
#nombre del archivo donde se imprime la traza
archivoTraza = "traza"
#nombre de la clase de la medida de distancia
distancia = DistanciaRNPC
```

A.2. Diseño

A continuación se presenta un el diseño derivado de los requisitos definidos en la sección anterior.

A.2.1. Diseño de componentes

El sistema se basa en tres componentes distintos, las relaciones entre estos componentes pueden observarse en el diagrama que aparece en la figura A.1.

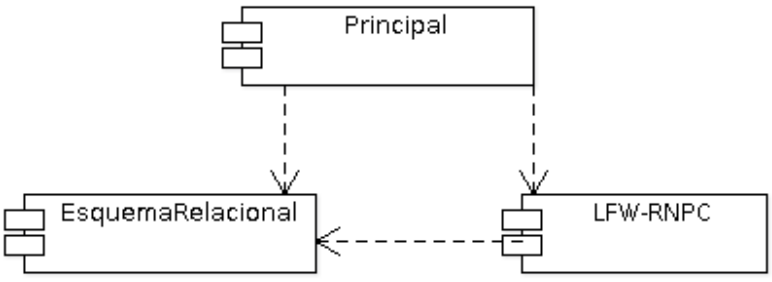


Figura A.1: Diagrama de componentes

- *EsquemaRelacional*:
  Este componente implementa las operaciones de gestión del esquema relacional, indización de instancias relacionales, de relaciones y de atributos.

- *Principal*:
  Este componente implementa las operaciones de entrada-salida (impresión de trazas, almacenado de ficheros de resultados, creación de folds para la evaluación y lectura de parámetros).

- *LFW-RNPC*:
  Dentro de este componente se encuentran las clases que implementan propiamente el algoritmo.

A.2.2. Diseño de clases

A continuación se exponen las clases de cada componente.

Componente EsquemaRelacional

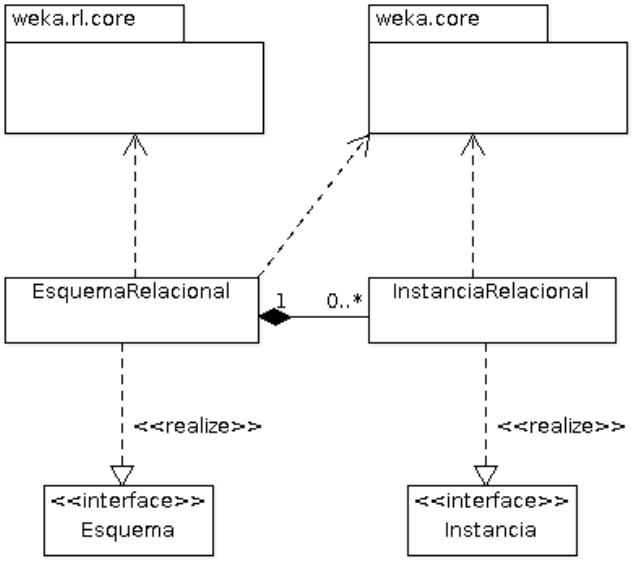


Figura A.2: Clases del componente EsquemaRelacional

- *Esquema*: Esta clase gestiona la información sobre el esquema relacional que define el dominio de aprendizaje. En la inicialización se componen todas las instancias relacionales, indizando las tuplas de las relaciones en función de los valores de las claves que referencian. También se indizan los atributos de cada relación por tipo.

- *Instancia*: Este interfaz abstrae las operaciones.
- *Instancia Relacional*: Esta clase implementa las operaciones para el acceso a las tuplas de cada relación que componen cada instancia relacional.

Componente LFW-RNPC

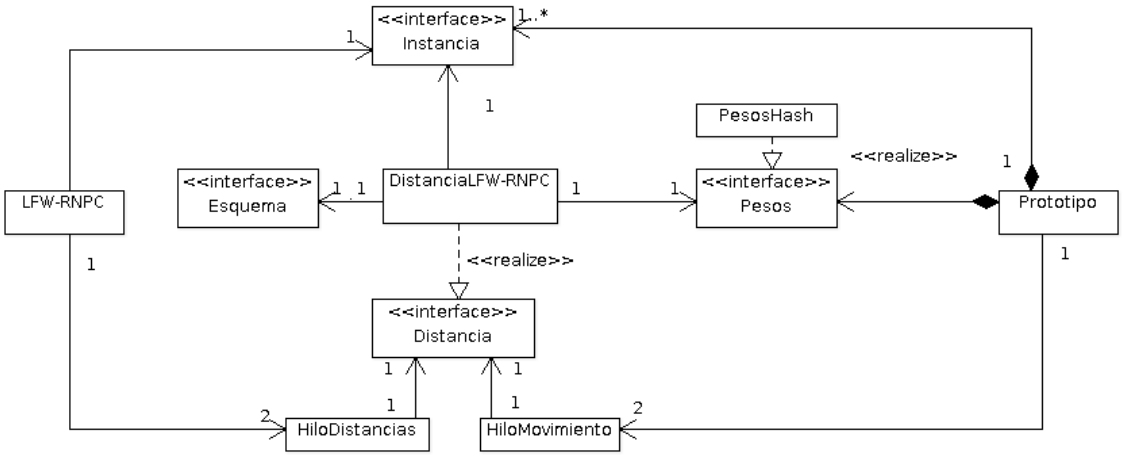
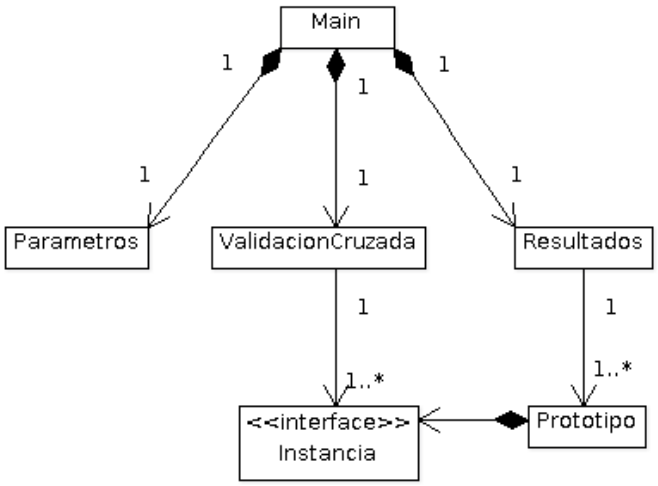


Figura A.3: Clases del componente LFW-RNPC

- *Algoritmo*: Esta clase implementa el algoritmo propiamente, incluyendo cada uno de los pasos descritos en la sección 3.1.
- *Prototipo*: Esta clase implementa las operaciones de gestión de instancias de la región, elección de representantes y cálculo de características de cada prototipo.
- *Pesos*: Esta interfaz abstrae las operaciones de obtención y actualización de los pesos asociados a cada atributo de cada relación del dominio, ocultando las estructuras de datos para almacenar y modificar estos pesos.
- *PesosHash*: Esta clase implementa los servicios de actualización y gestión de los pesos. Utiliza tablas *Hash* para acceder a los pesos de una región.
- *Distancia*: Este interfaz abstrae el cálculo de instancias. Este interfaz se ha añadido para facilitar el cambio de la distancia utilizada en el algoritmo.
- *DistanciaLFW-RNPC*: Esta clase implementa la distancia descrita en la sección 3.2.

- *HiloMovimiento*: Esta clase implementa el hilo de ejecución el cálculo del representante de cada región.
- *HiloDistancias*: Esta clase implementa el hilo de ejecución para el cálculo de los prototipos más cercanos.

Componente Principal



cmcm

Figura A.4: Clases del componente principal

- *Main*: Esta clase engloba las operaciones para ejecución del algortimo y coordinación de las operaciones de entrada y salida.
- *ValidacionCruzada*: Esta clase implementa la división de instancias en cubetas para la validación cruzada.
- *Parámetros*: Esta clase gestiona la lectura de parámetros y su verificación.
- *Resultados*: Gestiona el guardado y procesado de los resultados obtenidos en cada ejecución.

Anexo B

Gestión del proyecto

B.1. Planificación

A continuación se muestra el diagrama de Gantt que comprende la planificación de la consecución de las distintas fases del proyecto determinadas por los objetivos descritos en la sección 1.3.

Entre los participantes se cuenta con un Jefe de Proyecto (JP), un Analista (AN), un investigador (IN) y un programador (PR). El tiempo de cada tarea (T) se cuenta en semanas dedicadas.

Tarea	Inicio	Fin	T	Participantes
Diseño e implementación de LFW-RNPC	**01/09/09**	**27/01/10**	**20**	**JP,AN,IN,PR**
Estado de la cuestión	01/09/09	29/10/09	8	JP,AN,IN,PR
Estudio de Weka, RelWEKA,RNPC	21/09/09	16/10/09	3	JP,AN,IN,PR
Diseño de distancia	12/10/09	18/10/09	5	JP,IN
Diseño de actualización de pesos	19/10/09	18/10/09	4	JP,IN
Diseño de componentes y clases	19/10/09	23/10/09	5	JP, AN
Implementación	09/10/09	27/01/10	9	JP,PR
Validación y verificación	9/10/09	23/03/10	16	JP, IN,PR
Experimentación y evaluación	**9/10/09**	**18/06/10**	**23**	**JP,AN,IN**
Definición y selección de prueba	9/10/09	16/01/10	10	JP,IN
Experimentación	8/02/10	14/06/10	12	JP,IN
Análisis de resultados	15/03/10	18/06/10	9	JP,IN
Redacción de la memoria	**24/05/10**	**10/09/10**	**11**	**JP,AN,IN**

Tabla B.1: Tareas del proyecto

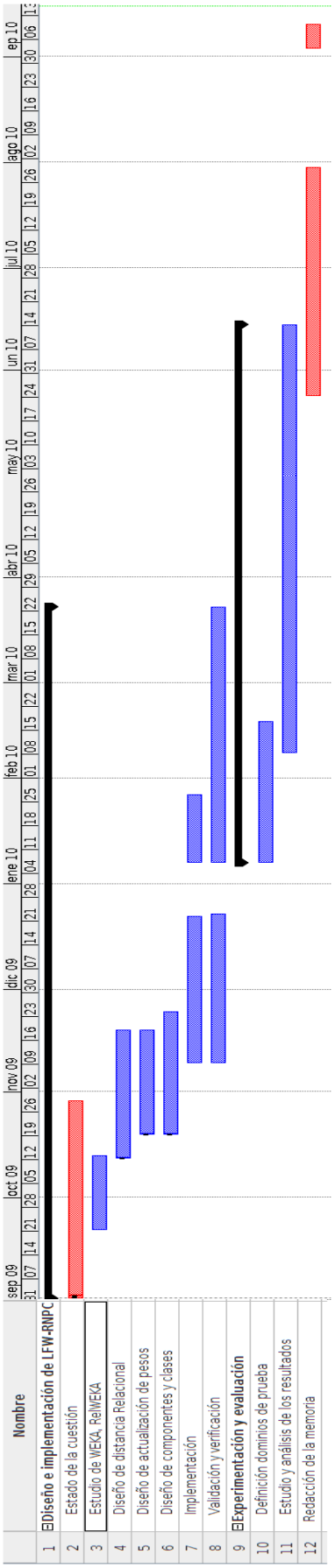


Figura B.1: Diagrama de Gantt

B.2. Recursos Humanos

A continuación se describen los costes asociados al personal participante en el proyecto de acuerdo a las tarifas estipuladas en el BOE 063, de 13/03/2008, sección 3, páginas 15242 a 15243. Se considera el importe bruto para cada participante.

Participante	**Horas**	**€/Hora**	**Total**
JP	70	35	2450
AN	130	33	4290
IN	250	33	8250
PR	240	29	6960
Total			**21950**

Tabla B.2: Recursos Humanos

B.3. Recursos materiales

Se enumeran los rescursos materiales consumidos en el desarollo del proyecto, teniendo en cuenta la devaluación e impuestos pertinentes sobre los precios que se muestran. Además se señala directamente el coste total que ha supuesto cada artículo al proyecto, es decir, se han tenido en cuenta todas la unidades requeridas:

Recurso	**Precio final (€)**
Ordenador para experimentación	1100
Memoria USB	10
Material de oficina	15
Impresora de inyección	30
Cartuchos de tinta	20
Total	**1175**

Tabla B.3: Recursos materiales

Bibliografía

[1] Dzeroski S., Lavrac N.: Relational Data Mining. Springer. 2001.

[2] Fernández F., Isasi P.: Evolutionary design of nearest prototype classifiers. Journal of Heuristics. 2004.

[3] Fernández F., Isasi P.: Local feature weighting in Nearest Prototype Classification. IEEE transactions on neural networks. 2007.

[4] García-Durán R., Fernández F., Borrajo D.: Prototypes Based Relational Learning. Lecture Notes in Computer Science. Springer. 2008.

[5] García-Durán R., Fernández F., Borrajo D.: REPLICA: Relational Policies Learning in Planning CAEPIA 2007, Workshop Planning, Scheduling and Constraint Satisfaction. Salamanca, Spain. November 2007.

[6] Kirsten M., Wrobel S., Horváth, T.: Distance Based Approaches to Relational Learning and Clustering. Relational Data Mining. Springer. 2001.

[7] Woznica A., Kalousis A., Hilario M.: Distance-based learning over extended relational algebra structures. Proceedings of the 15th International Conference of Inductive Logic Programming. 2005.

[8] McLachlan GJ., Do K.A., Ambroise C.: Analyzing microarray gene expression data. Wiley. 2004.

[9] Mithchell Tom M.: Machine Learning. WBC-McGraw-Hill. 1997.

[10] De Raedt. L.: Attribute-value learning versus inductive logic programming: The missing links. Proceedings of the Eights International Conference on Inductive Logic Programming. Springer.

[11] Emde W., Wettschereck D.: Relational Instance-Based Learning. 1996.

[12] A. Kalousis, A. Woznica, and M. Hilario. A unifying framework for relational distance-based learning. Technical report, Department of Computer Science, University of Geneva. 2005.

[13] Cohen W., Ravikumar P., Fienberg S.: A Comparison of String Distance Metrics for Name-Matching Tasks. American Association for Artificial Intelligence. 2003.

[14] Zhang T.: Leave-One-Out Bounds for Kernel Methods. Neural Computation. Massachusetts Institute of Technology. 2003.

[15] Somervuo P., Kohonen t.: Self-Organizing Maps and Learning Vector Quantization for Feature Sequences. 2004.

[16] De la Rosa, T.m García-Olaya, A.; and Borrajo, D.: Using cases utility for heuristic planning improvement. In Weber, R., and Richter, M., eds., Case-Based Reasoning Research and Development: Proceedings of the 7th International Conference on Case-Based Reasoning, volume 4626 of Lecture Notes on Artificial Intelligence, 137-148. Belfast, Northern Ireland, UK: Springer Verlag. 2007.

[17] Witten I., Frank E.: Data Mining: Practical Machine Learning Tools and Techniques,2nd Edition, Morgan Kaufmann,2005

[18] Blockeel, H., De Raedt, L.: Top-down induction of first order logical decision trees. Artificial Intelligence 101(1-2). 1998.